Math. Struct. in Comp. Science: *page* 1 of 49. © Cambridge University Press 2016
doi:10.1017/S0960129516000219

Computations with oracles that measure vanishing quantities

EDWIN BEGGS[†], JOSÉ FÉLIX COSTA[‡], DIOGO POÇAS[§] and JOHN V. TUCKER[†]

[†]*College of Science, Swansea University, Swansea, SA2 8PP, Wales, U.K.*
Emails: e.j.beggs@swansea.ac.uk, j.v.tucker@swansea.ac.uk
[‡]*Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal*
Email: fgc@math.ist.utl.pt
[§]*Department of Mathematics and Statistics, McMaster University,*
Hamilton, Ontario, L8S 4K1, Canada
Email: pocasd@math.mcmaster.ca

Received 31 January 2014; revised 20 September 2015

We consider computation with real numbers that arise through a process of physical measurement. We have developed a theory in which physical experiments that measure quantities can be used as oracles to algorithms and we have begun to classify the computational power of various forms of experiment using non-uniform complexity classes. Earlier, in Beggs et al. (2014 *Reviews of Symbolic Logic* **7** (4) 618–646), we observed that measurement can be viewed as a process of comparing a rational number z – a test quantity – with a real number y – an unknown quantity; each oracle call performs such a comparison. Experiments can then be classified into three categories, that correspond with being able to return test results

$$\begin{aligned} z < y \text{ or } z > y \text{ or } &\textit{timeout},\\ z < y \text{ or } &\textit{timeout},\\ z \neq y \text{ or } &\textit{timeout}. \end{aligned}$$

These categories are called *two-sided*, *threshold* and *vanishing experiments*, respectively. The iterative process of comparing generates a real number y. The computational power of two-sided and threshold experiments were analysed in several papers, including Beggs et al. (2008 *Proceedings of the Royal Society, Series A (Mathematical, Physical and Engineering Sciences)* **464** (2098) 2777–2801), Beggs et al. (2009 *Proceedings of the Royal Society, Series A (Mathematical, Physical and Engineering Sciences)* **465** (2105) 1453–1465), Beggs et al. (2013a *Unconventional Computation and Natural Computation (UCNC 2013)*, Springer-Verlag 6–18), Beggs et al. (2010b *Mathematical Structures in Computer Science* **20** (06) 1019–1050) and Beggs et al. (2014 *Reviews of Symbolic Logic*, 7 (4):618-646). In this paper, we attack the subtle problem of measuring physical quantities that vanish in some experimental conditions (e.g., Brewster's angle in optics). We analyse in detail a simple generic vanishing experiment for measuring mass and develop general techniques based on parallel experiments, statistical analysis and timing notions that enable us to prove lower and upper bounds for its computational power in different variants. We end with a

comparison of various results for all three forms of experiments and a suitable postulate for computation involving analogue inputs that breaks the Church–Turing barrier.

1. Introduction

Computation theory is about data, algorithms, programs and machines. When the data are real numbers, our computations operate with approximations: typically, a real number y that is given as a sequence

$$z_0, z_1, z_2, \ldots, z_k, \ldots$$

of rational numbers that are approximations to y. Indeed, the real number y is defined as an idealisation of a *process* of rational approximation in the form of an equivalence class of rational Cauchy sequences. Finite computations with the infinite representations of y operate with the finite representations of rational numbers $z_0, z_1, z_2, \ldots, z_k, \ldots$ together with a modulus of convergence.[†] Our computability theories are not concerned with where the real numbers come from: the intuitions and abstractions of real number computation form a world of their own. This is entirely satisfactory if the real number comes from another computation. However, let us expand this world somewhat by asking the question:

Suppose a real number y that enters a computation comes from a process of physical measurement. How can we model this process? What effect does measurement have on our theory of real number computation?

Whilst logical aspects of measurement have been theorised – see the magnum opus (Krantz et al. 1990) – computable aspects of measurement have been rather neglected. An early analysis on the computability of physical constants is Geroch and Hartle (1986), where an informal attempt at defining measurement can be found. We are developing a new theory that combines measurement and computation, in a series of papers that examines a range of experiments and their effects on computation (Beggs et al. 2008, 2009, 2010a,b,c, 2012a,b, 2014). Like Geroche and Hartle, our study is rooted in physics but it uses a methodology that combines physical theories with the rich concepts and results of computability and complexity theory. An abstract study of measurement, tailored to computable analysis, is Pauly (2009); Pauly uses the representation methods characteristic of the type 2 computability models – e.g., after Weihrauch (2000) – to model an interface to the data extracted from physical behaviour.

In this paper, we will be able to reach some preliminary conclusions by solving a remaining subtle problem, that of measuring physical quantities that vanish in some experimental conditions.

What is measurement? The essence of any measurement is a comparison: see Hempel (1952) where comparisons are based on events in the experimental setup. Let the real number y be the value of the unknown quantity we wish to measure. Then, a comparison is made with a known quantity z; this quantity is a rational number because it describes the quantity using units of measurement (e.g., 3.75 means 3 whole units and $\frac{3}{4}$ of a unit). In running an experiment, one expects there to be a schedule that allocates reasonable

[†] The combination makes the approximation process computable.

waiting times. The actual experimental time is likely to be a function f of the difference between our known quantity and the unknown value, $f(|y-z|)$. This is evidenced by our study of many experiments and it has become a fundamental assumption in the theory. In many experiments, we expect *one* of the following values, where *timeout* indicates that the experimental time has exceeded the allocated schedule:

$$z < y \text{ or } z > y \text{ or } \textit{timeout}.$$

The process can be repeated with a new known quantity z', which is easily determined by the previous result. Thus, a sequence of measurements can be generated that approximate y. We have studied many such experiments and call them *two-sided experiments* because approximations can be found above or below y.

However, not all experiments are two-sided. In Beggs et al. (2013a,b), we studied experiments in which the unknown quantities were thresholds. In this case, the process produces one of these test results:

$$z < y \text{ or } \textit{timeout}.$$

We showed how the process can be repeated with a new known quantity z' determined by the previous result and a sequence of approximate measurements generated – but the case is complicated. We call these *threshold experiments.*

In this paper, we study a third form of experiment in which the unknown quantity vanishes. In this case, the process produces one of these test results:

$$z \neq y \text{ or } \textit{timeout}.$$

Thus, we know the unknown and known quantities are not equal but we do not know which is larger. It is harder to see how to repeat the process with a new known quantity z', using the result, in order to generate a sequence that measures y. Here, we give new techniques based upon running parallel experiments, statistical analysis and exploiting timing notions that enable us to generate a sequence and enable us to prove lower and upper bounds for its computational power in different variants. This done we are able to compare the various results for all three forms of experiment and draw some conclusions.

Our theory combines measurement and computation by integrating models of experiments with models of computation for the data obtained. At the heart of our theory is the idea that an experimenter measures a physical quantity by applying an experimental procedure to equipment and that an experimental procedure is an algorithm of some kind. We model this idea as follows:

The experimental procedure is modelled as a Turing machine. The equipment is modelled as a physical device whose behaviour is governed by a physical theory. The equipment is connected to the Turing machine as a physical oracle with a protocol to manage the interface.

The Turing machine abstracts the experimental procedure, encoding the experimental actions and data by a program; it also is able to process the data obtained by measurement. Thus, this mathematical approach captures:

i. the measurement process;
ii. the use of data from measurement in subsequent computations; and, indeed,
iii. arbitrary sequences of interactions with equipment.

The protocols that manage the interface between algorithm and experiment are sophisticated, controlling the precision and timings of queries to the oracle. The precision in the data can be (a) infinite, (b) finite and unbounded or (c) finite and fixed. The timings are limited, bounded by a function of the query. Finally, the computational power is analysed under the complexity constraint of polynomial time and expressed in terms of non-uniform complexity theory. Since non-uniform complexity seems to be little known, for convenience, we give a quick summary in Section 3.

The models of a physical systems are based upon some fragment of a physical theory; these fragments are parameters in all our applications for, in principle, changing the assumptions about a physical model can change the computational properties. This method of formalising the theory was seen as basic to any theoretical methodology seeking to explore the relationship between physical systems and computation; it was described in Beggs and Tucker (2006, 2007a). Martin Ziegler used the idea with great effect on the longstanding problem of defining a physical Church–Turing Thesis in Ziegler (2009).

In this paper, centre stage is taken by the new methods based on analysing experimental times. We use the measurement of the time to determine the computational power of a Turing machine using a typical vanishing experiment as an oracle. In the first method, we perform two experiments in parallel differing in only one parameter, and determine which experiment finishes first. In the second method, we time experiments by means of a clock. These techniques introduce the problem of precision into our thinking about time. We can ask: What is the smallest temporal resolution with which we can distinguish the order of two events? What is the spacing between the ticks of the clock, i.e., the temporal resolution which will be recorded as a different time on the clock output. The precision ε in time is not asymptotic – as the precision in the quantity (i.e., $\varepsilon \to 0$) – but rather a tolerance that can be quantified as a function of the query. Thus, here we consider two types of precision: precision in the time of oracle consultation and precision in the concept to be measured.[‡]

In a Type I vanishing experiment, two instances E and E' of an experiment are run with different values of a parameter (typically with one unknown value and one known value set by the experimenter). In our main example, this parameter is a mass. The result returned by the experiment is simply whether experiment E terminates before E', whether E' terminates before E, or otherwise (i.e., neither terminate in the allowed time, nor they are judged to have terminated simultaneously). *Time is compared rather than quantified.* The precision of the experiment determines how accurately the actual value of the parameter used in the experiment corresponds to the data in the query sent by the Turing machine. The fact that this is a vanishing experiment, rather than two-sided or threshold, dictates the algorithm.

[‡] We have discussed the case of precision of the quantity in several of our earlier papers. Here, we note that the problem of errors in experiments that cannot be made arbitrarily small just by putting more care and resources into the experiment is tackled by repeating experiments to obtain statistical data, from which we obtain probabilistic complexity classes.

Theorem 1.1. The following are lower and upper bound results for Type I parallel methods for the three kinds of precision:

1. If a set A is decided in polynomial time by a deterministic oracle Turing machine coupled with a Type I vanishing quantity experiment of infinite or unbounded finite precisions, then $A \in P/poly$. If a set A is in $P/poly$, then A is decided by a deterministic oracle Turing machine coupled with a Type I vanishing quantity experiment of infinite or unbounded finite precisions.
2. If a set A is decided in polynomial time by an oracle Turing machine coupled with a vanishing quantity experiment of fixed finite precision, then $A \in BPP//log\star$. If a set A is in $BPP//log\star$, then A is decided in polynomial time by an oracle Turing machine coupled with a vanishing quantity experiment of fixed finite precision.

Note that, although lower and upper bounds of a Type I vanishing experiment coincide in each case, they are stated separately in order to compare them with the corresponding statements of a Type II vanishing experiment (described below), where the lower and upper bounds do not coincide in all the cases considered.

In a Type II vanishing experiment, an instance of an experiment is run with a parameter determined by data in a query from the Turing machine, up to a certain precision. The query is timed by the system clock, and an answer to the query is returned. The system clock is taken to be incremented by integers, so the time returned actually states that the experiment terminated between two 'ticks' of the system clock, thus introducing a time resolution. A priori, we are not concerned with any 'physical time' for the experiment, just the timing of termination measured by the system clock.

Theorem 1.2. The following are lower and upper bound results for Type II clock methods for the three kinds of precision:

1. If a set A is decided in polynomial time by a deterministic oracle Turing machine coupled with a Type II vanishing quantity experiment of infinite precision, then $A \in P/poly$. If a set A is in $P/log\star$, then A is decided by a oracle Turing machine in polynomial time coupled with a vanishing quantity experiment of infinite precision.
2. If a set A is decided in polynomial time by a deterministic oracle Turing machine coupled with a Type II vanishing quantity experiment of unbounded finite precision, then $A \in P/poly$. If a set A is decided in polynomial time by a deterministic oracle Turing machine coupled with a Type II vanishing quantity experiment of unbounded finite precision and exponential protocol, then $A \in BPP//log\star$. If a set A is in $BPP//log\star$, then A is decided by a oracle Turing machine in polynomial time coupled with a vanishing quantity experiment of unbounded finite precision.
3. If a set A is decided by a deterministic oracle Turing machine coupled with a Type II vanishing quantity experiment of fixed finite precision, then $A \in BPP//log\star$. If a set A is in $BPP//log\star$, then A is decided in polynomial time by a oracle Turing machine coupled with a vanishing quantity experiment of fixed precision.

Vanishing oracles are new and the methods for two-sided and threshold oracles do not apply to them – this has led us to discover the use of timing, which raises subtle points for the theory. The upper bound known so far for the two-sided oracles with non-infinite precision is $P/poly$ (except for particular types of two-sided oracles considered in Beggs et al. (2009) and Beggs et al. (2012a) for which the upper bounds are $P/poly$ and $BPP//log\star$, respectively). The upper bound known so far for the threshold oracles with non-infinite precision is $BPP//log^2\star$ (Beggs et al. 2013a).

In Section 2, we introduce two vanishing experiments: the measurement of Brewster's angle in optics and a special beam balance for mass designed for our theoretical analysis, which we call the vanishing balance experiment (VBE for short). In Section 3, we define the relevant non-uniform complexity classes. In Sections 4 and 8, we discuss the operating protocols between Turing machines and stochastic oracles. In Sections 5–7, we will introduce the VBE machines together with measurement algorithms for the three types of precision. Finally, for each type of precision, we will characterise the complexity classes decided by such machines in polynomial time. Lower bounds are proved in Section 10 and upper bounds in the following Section 11.

2. Examples of vanishing quantity experiments

We now present two vanishing quantity experiments and adopt the second for our analysis. The first optical experiment shows that there are physical scenarios in which an experiment yields $z \neq y$ without yielding $z < y$ or $z > y$. The second beam balance experiment is artificial to some extent, but is a convenient to use in developing the theory: measuring mass by means of a beam balance is the simplest form of experimental comparison that is not trivial. Furthermore, each of our three types of experiment can be demonstrated by an adaptation of the beam balance, showing that the beam balance is a reassuringly primitive experiment, ideal for theory making.

Our methodology for any experiment begins with settling on a fragment of physical theory sufficient to specify a model of the equipment and its behaviour (Beggs and Tucker 2006, 2007a). However, in our examples, there are some simple choices for the assumptions.

2.1. *The Brewster angle experiment*

The Brewster Angle Experiment is an experiment based on the principles of classical optics. It was first suggested as an example of a vanishing value experiment in Beggs et al. (2010d). The book (Born and Wolf 1964) provides an useful reference for the experiment that we will now describe. When a plane wave falls on to a boundary between two homogeneous media, it is split into two: a transmitted wave propagating into the second medium and a reflected wave propagated back into the first medium. Figure 1 provides an illustration of this phenomenon.

The experimental apparatus consists of a surface dividing two media, a projector and a light detector. We can send a light beam into the surface with a certain incidence angle φ. The electric field is perpendicular to the direction of propagation and can be decomposed into components parallel (subscript $\|$) and perpendicular (subscript $\perp$) to the plane of incidence. Let $E_{i,\|}$ and $E_{i,\perp}$ denote the components of the incident ray, $E_{r,\|}$ and $E_{r,\perp}$ the components of the reflected ray, and $E_{t,\|}$ and $E_{t,\perp}$ the components of the transmitted ray.

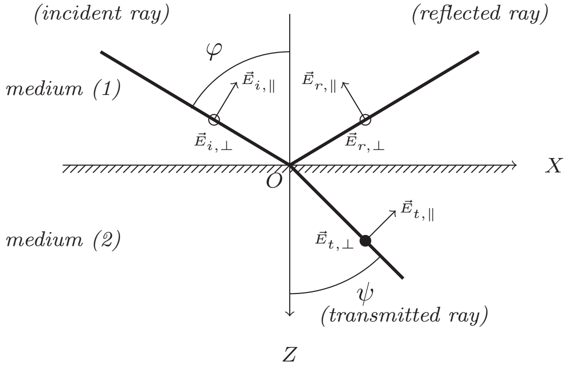


Fig. 1. Elements of incident, reflected and transmitted light rays. The indexes of the electrical field E denote the *incident*, i, *reflected*, r, and *transmitted*, t, rays, together with the *normal*, n, and the *parallel*, p, components of the field. The black circle denotes the normal component pointing forward and the white circle denotes the normal component pointing backwards.

Let φ be the angle of incidence and ψ be the angle of transmission. From optics, we can deduce several relations between the components of the electric field, called Fresnel formulas:

$$E_{t,\|} = \frac{2\sin\psi\cos\varphi}{\sin(\varphi+\psi)\cos(\varphi-\psi)}E_{i,\|}, \quad E_{t,\perp} = \frac{2\sin\varphi\cos\psi}{\sin(\varphi+\psi)}E_{i,\perp}\ ,$$

$$E_{r,\|} = \frac{\tan(\varphi-\psi)}{\tan(\varphi+\psi)}E_{i,\|}, \quad E_{r,\perp} = -\frac{\sin(\varphi-\psi)}{\sin(\varphi+\psi)}E_{i,\perp}\ .$$

Brewster's law states that*for some value of the angle of incidence, the reflected light is totally polarised in the direction normal to the plane of incidence.* From the Fresnel formulae, we see that it occurs when $\varphi + \psi = \frac{\pi}{2}$, so that $E_{r,\|} = 0$. Our objective is to measure the Brewster angle φ_B. For the sake of simplicity, we assume some scale such that $0 < \varphi_B < 1$. In this type of experiment, we cannot infer any information about the Brewster angle simply by sending a ray with desired angle ϕ. The reason is that, as φ approaches φ_B, the intensity of the electric field decreases to 0, either if $\varphi < \varphi_B$ or if $\varphi > \varphi_B$.

Consider that an instance of the experiment begins by sending a light beam, polarised in the horizontal direction, with angle incidence φ so that $E_{i,\perp} = 0$, so that the reflected ray is also polarised. Furthermore, as φ approaches φ_B, the reflected ray vanishes completely. Denote by $\mathbf{H}_r$ the magnetic field produced by the reflected ray. We know that $\mathbf{H}_r$ is perpendicular to $\mathbf{E}_r$ and to the direction of propagation of the reflected ray; this implies that $\mathbf{H}_r = \mathbf{H}_{r,\perp}$. Furthermore, we have the following relation between the amplitudes of the magnetic field and of the electric field: $H_r = nc\epsilon_0 E_r$, where n is the index of refraction of the first medium, c is the speed of light in the vacuum and ϵ_0 is the vacuum permittivity. We assume the existence of a light detector on the reflection side that absorbs the energy of the reflected ray. This detector can be a photovoltaic detector that reacts when it has absorbed energy above a threshold limit Ω. The directional energy flux of the reflected ray is then given by the Poynting vector, $\mathbf{S}_r = \mathbf{E}_r \times \mathbf{H}_r$, with the average magnitude of $\langle S\rangle = nc\epsilon_0 E_r^2/2$. Finally, let α be the cross-section area of the light beam. The time T_{exp}

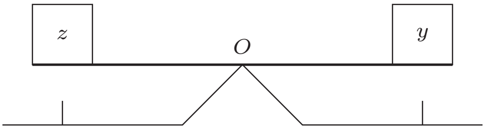


Fig. 2. Schematic depiction of the *vanishing balance experiment*.

taken for the light detector to absorb the threshold energy is given by

$$T_{\text{exp}} = \frac{\Omega}{\alpha\langle S\rangle} = \frac{2\Omega}{n\alpha c\epsilon_0 {E_r}^2}.$$

Even though the above expression seems very detailed, the important point is that the experimental time is proportional to the inverse of the square of the electric field of the reflected ray. Thus, we get the following experimental time, T_{exp}, given in some abstract units,

$$T_{\text{exp}}(z,\psi) = \frac{\tan(\varphi+\psi)^2}{\tan(\varphi-\psi)^2}\ .$$

Reviewing the assumptions we have made, we have made an easy application of the classical rules of optics. If we were to replace classical by quantum optics, we would have the more subtle matter of having the expected time to detecting a certain number of photons of a given frequency.

2.2. *The vanishing balance experiment (VBE)*

The following experiment (depicted in Figure 2) is a variation of the balance scale for mass. The balance has two pans with a pressure stick below each pan. On the right pan, there is a body with the unknown mass of size y. To measure y, we place a test mass z on the left pan. If $z = y$, then the scale will not move since the lever is in equilibrium. But, if $z \neq y$, then one of the pans will move down and sooner or later it will press one of the pressure sticks. However, when $z \neq y$, there is *no information about which of the pans sank*, only that one of them did.

There are also other assumptions that can be made explicit about the experiment: (a) y is a real number in $[0,1]$; (b) the mass z can be set to any dyadic rational in the interval $[0,1]$; (c) a pressure-sensitive stick is placed below each side of the balance, such that, when one of the pans touches the pressure-sensitive stick, it reacts producing a signal; (d) the mass z can be set so that the procedure starts from absolute rest; (e) the friction between the masses and the pans is large enough so that these will not slide away from their original position once the scale is in motion and (f) the bar on which the masses are placed is made of an homogeneous material, so that the two pans have exactly the same weight. Assuming that the test mass weighs z and the unknown mass weighs y, the cost of the experiment, $T_{\text{exp}}(z,y)$, which is the time taken for one of the pans to touch the pressure stick, in some abstract units of time, is given by

$$T_{\text{exp}}(z,y) = \sqrt{\frac{z+y}{|z-y|}}. \tag{1}$$

This expression for the time, which exhibits an exponential growth on the precision of z with respect to the unknown y, is typical in physical experiments, regardless of the concept being measured.

Reviewing our assumptions again, we have used the standard theory of mechanics of rigid bodies. This means that we have neglected molecular effects, such as Coulomb adhesion–friction, which would alter the behaviour of the system. However, the purpose of using this balance experiment is to have a generic example of a vanishing experiment, just as in other places, we had generic examples of threshold and two-sided experiments.

Note as $y \to z$, we are taking a limit of a theory of physics, and in practice another theory may intervene, e.g., quantum mechanics. However, the case for introducing quantum theory might be regarded as similar to that of imposing a general relativistic limit on the size of a Turing machine!

3. Physical oracles and their computational power

3.1. *Turing machines and physical oracles*

In computability theory, after Turing and Post, an oracle for a Turing machine is accessed by the Turing machine, which answers queries from a query tape about the membership of elements of a set; the responses take one time step, and have no error.

A *physical oracle* answers queries from a query tape about the behaviour of a physical process. In contrast to a set-theoretic oracle, it may take a variable amount of time to answer a query – if indeed it does – and may give error prone results. The Turing machine is connected to physical reality via a protocol. To prevent problems like the Turing machine waiting an arbitrarily long time for a response to a query, the protocols will impose a limit on waiting times.

The combination of the Turing machine and physical oracle may be thought of as having two possible functions. First, as in computability theory, information from the physical oracle – physical observations – may be used to boost the computational power of the Turing machine. Second, an innovation, the Turing machine may be used to control some aspect of reality, such as directing a sequence of steps in an experiment or controlling a hybrid system. In this paper, we will be concerned with the first function.

The architecture is illustrated in the diagram of Figure 3.

Consider a Turing machine running a program that queries a physical oracle via a protocol or interface. What does the Turing machine (or its programmer) actually know? The physical experiment itself is beyond its knowledge, it only sees queries and replies. To study what such an augmented Turing machine can compute, we can give an axiomatic specification of the physical oracle, which involves a segment of physical theory and a specification of the particular physical system that serves as the oracle.

The mathematical details of construction and operation of a Turing machine with a physical oracle, and the key notions of precision, waiting times, timeouts, etc., can be found in the early papers of our series such as Beggs et al. (2008, 2009, 2012b).

One way of studying what a Turing machine with physical oracle can compute in a given time is to use non-uniform complexity theory.

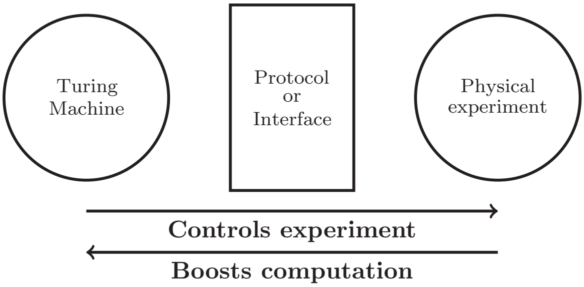


Fig. 3. The architecture of a Turing machine with a physical oracle.

3.2. *Physical oracles and non-uniform complexity classes*

We consider the classical word acceptance problem for a Turing machine, but incorporating advice functions: we refer to Balcázar et al. (1995) for advice and non-uniform complexity, only saying what is required for understanding this paper.

By an *advice function*, we mean any total map $f : \mathbb{N} \to \Sigma^*$, returning finite words in an alphabet Σ; in our case, Σ is just the binary alphabet $\{0, 1\}$. The *pairing function* is the well-known map $\langle -, - \rangle : \Sigma^* \times \Sigma^* \to \Sigma^*$, computable in linear time, that allows us to encode two words in a single word over the same alphabet by duplicating bits and inserting a separation symbol ‘01.’

The definition of *non-uniform complexity class* follows:

Definition 3.1. Let $\mathcal{B}$ be a class of sets and $\mathcal{F}$ be a class of advice functions. Define the class $\mathcal{B}/\mathcal{F}$ as the class of sets A such that there exists a set $B \in \mathcal{B}$ and an advice $f \in \mathcal{F}$ such that, for every word $x \in \Sigma^*$, $x \in A$ if, and only if, $\langle x, f(|x|) \rangle \in B$.

Take the class $\mathcal{B}$ to be the class P of sets decidable by Turing machines in polynomial time. Now, we have to choose the class of advice functions $\mathcal{F}$. Note that the advice functions are not, in general, computable but the bounds on length are computable; e.g., in the advice class *poly*, any advice function $f : \mathbb{N} \to \Sigma^*$ is bounded by a (computable) polynomial p such that, for all $n \in \mathbb{N}$, $|f(n)| \leqslant p(n)$. Thus, we define $P/poly$ and, similarly, P/log with the advice class *log*.

A prefix non-uniform complexity class uses only prefix advice functions, i.e., functions f such that $f(n)$ is always a prefix of $f(n + 1)$. The idea behind prefix non-uniform complexity classes is that the advice given for inputs of size n may also be used to decide smaller inputs.

Definition 3.2. Let $\mathcal{B}$ be a class of sets and $\mathcal{F}$ a class of advice functions. The prefix advice class $\mathcal{B}/\mathcal{F}\star$ is the class of sets A for which some $B \in \mathcal{B}$ and some prefix function $f \in \mathcal{F}$ are such that, for every length n and input w, with $|w| \leqslant n$, $w \in A$ if, and only if, $\langle w, f(n) \rangle \in B$.

Note that if *exp* is the set of advice functions bounded in size by functions in the class $2^{O(n)}$, then P/exp contains *all* sets. However, even a much smaller choice of $\mathcal{F}$ allows classically non-computable results: The *Halting Set*

$$H = \{0^n : \text{Turing machine with code } n \text{ halts on } 0\}$$

is in $P/log\star \subset P/poly$.

For probabilistic complexity classes, we use the basic class BPP, which is word acceptance problems solvable with *bounded error probability* in polynomial time by a Turing machine with access to a fair independent coin toss oracle (this might itself be considered as a form of physical oracle). Here, bounded error probability means that there is a number $\gamma < \frac{1}{2}$, such that the error probability of a Turing machine for any input w is smaller than γ.

For the probabilistic complexity classes, there is a complication in the definition of $BPP/\mathcal{F}$. Notice that by requiring a set $B \in BPP$ and a function $f \in \mathcal{F}$ such that $w \in A$ if, and only if, $\langle w, f(|w|)\rangle \in B$, we are demanding a fixed probability $\frac{1}{2} + \varepsilon$, $0 < \varepsilon < \frac{1}{2}$ (fixed by the Turing machine chosen to witness that $B \in BPP$) *for any possible advice*, instead of the more intuitive idea that the error $\gamma = \frac{1}{2} - \varepsilon$ only has to be bounded *after choosing the correct advice*. This leads to new definitions of the form $BPP//\mathcal{F}$ for the complexity class BPP; note the modest notational distinction, the double slash, that we will be using throughout this paper:

Definition 3.3. $BPP//poly$ is the class of sets A for which a probabilistic Turing machine $\mathcal{M}$ clocked in polynomial time, a function $f \in poly$, and a constant $0 < \gamma < \frac{1}{2}$ exist such that $\mathcal{M}$ rejects $\langle w, f(|w|)\rangle$ with probability at most γ if $w \in A$ and accepts $\langle w, f(|w|)\rangle$ with probability at most γ if $w \notin A$.

Definition 3.4. $BPP//log\star$ is the class of sets B for which a prefix advice function $f \in \log$, a probabilistic Turing machine $\mathcal{M}$ and a constant $\gamma < 1/2$ exist such that, for every word w with size less or equal to n, $\mathcal{M}$ rejects $\langle w, f(|w|)\rangle$ with probability at most γ if $w \in B$ and accepts $\langle w, f(|w|)\rangle$ with probability at most γ if $w \notin B$.

It is known that $BPP//poly = BPP/poly$, but it is not known whether the similar inclusion $BPP//log\star \subseteq BPP/log\star$ holds.

4. Protocols and the vanishing balance experiment

How do we make the connection between the digital computer (modelled as a Turing machine) and the analogue device (modelled as an oracle)? The arguments so far developed – such as in Beggs et al. (2008, 2010b,c, 2012b) – do not differ substantially from the classical analogue–digital protocol that we can find in books on hybrid computation, e.g., Bekey and Karplus (1968).

The main device for the transference of data from the digital component to the analogue component is a *query tape*. However, we have been working in a situation where the analogue oracle device furnishes to the digital computer two bits of information: YES/FIRST/LEFT, NO/SECOND/RIGHT, TIMEOUT and possibly INDISTINGUISHABLE. This is a restriction to the general analogue–digital converter (as in Bekey and Karplus (1968)), but it makes our theory closer to the realisability of hybrid machines: an answer tape is not needed and the result of the consultation of the oracle is encoded immediately after in the resulting state of the Turing machine.

The protocols that we will adopt for the vanishing experiments will be different from the protocols considered in previous papers. It seems that performing one instance of the experiment does not give much information about the relationship between the test mass z and the unknown mass y.[§]

We have to consider two instances of the experiment instead of one, with two different dyadic rationals, z_1 and z_2 and their respective experimental times $T_{\mathrm{exp}}(z_1, y)$ and $T_{\mathrm{exp}}(z_2, y)$. Now, suppose that we can determine which of the instances of the experiment ends first. That is, suppose that, for any two dyadic rationals, we can determine whether $T_{\mathrm{exp}}(z_1, y) < T_{\mathrm{exp}}(z_2, y)$, $T_{\mathrm{exp}}(z_1, y) = T_{\mathrm{exp}}(z_2, y)$, or $T_{\mathrm{exp}}(z_1, y) > T_{\mathrm{exp}}(z_2, y)$. Then, we could also determine, for a finite increasing sequence $z_1 < z_2 < \ldots < z_n$, which of the z_i corresponds to the instance that ends last. This would then imply something about y, thanks to the simple fact that y should be closer to dyadic rationals that consume more experimental time. This conclusion is a consequence of the fact that T_{exp} is increasing in the interval $[0, y)$ and decreasing in the interval $(y, 1]$. Now, we ponder on the assumption we made: given two dyadic rationals z_1 and z_2, corresponding to different instances of the experiment, how can we determine which of the instances ends last? There are two possible implementations of the experiment that can answer the question:

— To perform two experiments simultaneously, that is, to use two copies of the balance with the same unknown mass y in the right pan. We can place masses z_1 and z_2 at the left pans of the balances and start both experiments at the same time. If $T_{\mathrm{exp}}(z_1, y) < T_{\mathrm{exp}}(z_2, y)$, then the experiment with test mass z_1 sends a first signal and if $T_{\mathrm{exp}}(z_1, y) > T_{\mathrm{exp}}(z_2, y)$, then the experiment with test mass z_1 calls back first.

— Suppose we only have one balance, but now we can count the machine steps during an experiment until the end. In this way, we can begin by performing an instance of the experiment for test mass z_1, and counting the number T_1 of machine transitions that the experiment takes. Then, repeat the experiment for test mass z_2, obtaining a number T_2 of machine transitions. Finally, compare T_1 and T_2. If $T_1 < T_2$, then we conclude that $T_{\mathrm{exp}}(z_1) < T_{\mathrm{exp}}(z_2)$; if $T_1 > T_2$, then $T_{\mathrm{exp}}(z_1) > T_{\mathrm{exp}}(z_2)$.

The first solution overlooks a simple practical aspect. We are basically attempting to decide which of two events occurs first, but can we actually do it if the difference in times becomes very small? We could answer this question in the negative, and argue that there is a minimum time gap below which two events may appear simultaneous, so that we cannot tell which of them happens first. The second solution also introduces a problem. First, recall that we should set a bound on the time that we consider acceptable to wait for a response, the time schedule concept. If the experimental time exceeds the time schedule, then the count of steps and the experiment should be interrupted. This means that, when

[§] When we were studying two-sided experiments in, say, Beggs et al. (2012a), we saw that result 'LEFT' would imply that $z < y$ and result 'RIGHT' would imply that $z > y$. Also, when we were studying threshold experiments in Beggs et al. (2013a,b), we saw that result 'YES' would imply that $z > y$. But now, with vanishing experiments, we only have result 'YES' and this only implies that $z \neq y$. Of course, 'TIMEOUT' should occur very rarely, and may even not occur at all for some choices of unknown mass and time schedule.

```
PROTOCOL "COMPARE[1, $IP$]($z_1, z_2$)"
"MASS": Infinite precision case

    Receive as input the binary description of two dyadic rationals $z_1$ and $z_2$ of size $n$
        (possibly padded with 0s);
    Place a mass $z_1$ in the left pan of the first balance;
    Place a mass $z_2$ in the left pan of the second balance;
    Start both experiments at the same time;
    wait $T(n)$ units of time;
    Check which pressure stick have sent a signal first:
        if the first balance calls back first then return "FIRST";
        if the second balance calls back first then return "SECOND";
        if neither instance calls back, then return "TIMEOUT".
```

Fig. 4. Procedure that describes the VBE with the first implementation and infinite precision, for some unknown mass of size y and some time schedule T.

performing two instances of the experiment, any of them may result in a timeout. If one experiment times out and the other does not, we can still decide which of them ends first; nothing can be said when both experiments time out. However, it is not a great deal to solve this problem, since we can in principle increase the time schedule (padding the query z with 0s) until one of the experiments ends. There is another subtler situation in which we cannot decide which of the instances takes more time, if the number of machine transitions *is the same*, that is, when $T_1 = T_2$. In this situation, the two instances are indistinguishable. Increasing the time schedule will not help, nor will padding 0s.

We shall use these assumptions:

— In the first implementation, we assume that we can in fact distinguish the two events from one another. (This is, as we said, not feasible, but we are willing to consider it because it will provide us with some interesting results later on.) In this way, when we perform two instances of the experiment, there are three possible results: the first instance ends first; the second instance ends first; or both instances time out;

— In the second implementation, we assume that it is possible for two experiments to consume the same number of machine steps. In this way, when we perform two instances of the experiment, there are four possible results: the first instance ends first; the second instance ends first; both instances time out; or we could not decide which instance ends first (although none of them times out).

We must also take into account the imprecision associated with placing a test mass in the left pan. We do this in the same way as we have done in the previous papers (e.g., Beggs et al. (2008, 2012a)), by considering three types of precision:

a. *infinite precision* (see Figures 4 and 5): when the dyadic z is read in the query tape, a test mass z is simultaneously placed in the left pan,

b. *unbounded precision* (see Figures 4 and 5): when the dyadic z is read in the query tape, a test mass z' is simultaneously placed in the left pan such that $z - 2^{-|z|} \leqslant z' \leqslant z + 2^{-|z|}$ and

```
PROTOCOL "COMPARE$[2, IP, g](z_1, z_2)$"
"MASS": Infinite precision case

    Receive as input the description of two dyadic rationals $z_1$ and $z_2$ of size $n$
        (possibly padded with 0s);
    Place a mass $z_1$ in the left pan;
    wait $T(n)$ units of time, while
        counting the number of steps before receiving a signal from a pressure stick, $T_1$;
    if the experiment does not call back, then set $T_1 := T(n) + 1$;
    Place a mass $z_2$ in the left pan;
    wait $T(n)$ units of time,
        while counting the number of steps before receiving a signal from a pressure stick, $T_2$;
    if the experiment does not call back, set $T_2 := T(n) + 1$;
    Compare $T_1$ and $T_2$:
        if $T_1 < T_2$, then return "FIRST";
        if $T_1 > T_2$, then return "SECOND";
        if $T_1 = T_2 > T(n)$, then return "TIMEOUT".
        if $T_1 = T_2 \leq T(n)$, then return "INDISTINGUISHABLE".
```

Fig. 5. Procedure that describes the VBE with the second implementation and infinite precision, for some unknown mass of size y and some time schedule T.

c. *fixed precision* $\epsilon > 0$ (to be discussed later on): when the dyadic z is read in the query tape, a test mass z' is simultaneously placed in the left pan such that $z - \epsilon \leqslant z' \leqslant z + \epsilon$.

In the second implementation, there is also a different notion of imprecision, that appears when we count the number of machine transitions while the oracle is being consulted. We are making also the assumption that *all machine transitions take the same amount of physical time*. (Note that we really only require an order preserving 1–1 correspondence between physical and experimental time.) However, this is not necessarily true. This means that, when we count T_1 machine transitions, the actual time taken for the experiment may not be in $(T_1 - 1, T_1]$. To formalise this consideration, we define a new kind of imprecision, now related with time:

d. *time precision* g, given a map $g : \mathbb{N} \to \mathbb{N}$: when an experiment settled for the query word z takes an amount of time t, the number of machine transitions counted is T_1, where T_1 is a natural number uniformly sampled in $[\lceil t \rceil - g(|z|), \lceil t \rceil + g(|z|)]$.

Good examples for g are the following: $g(n) = 0$, i.e., full precision and we get back the assumption that all machine transitions take the same amount of time; $g(n) = c$, i.e., constant time precision; $g(n) = cn^k$, i.e., polynomial time precision; and $g(n) = c2^{kn}$, i.e., exponential time precision.

A first look at the protocol of Figure 4 may induce the reader to believe that the vanishing experiment can be in some way reduced to experiments analysed in previous papers such as the mass experiment in Beggs et al. (2010b), or the scatter experiment in Beggs et al. (2010c). However, this is not so: although the scheduling procedure is the same, in the vanishing experiment of first type, we are determining the order of events, regardless of how small the temporal separation (i.e., with infinite precision), and this is the only information we use.

Thus, we have to consider six different types of protocols, three for the first implementation and three for the second implementation. However, there are more than six possible protocols; observe that, after choosing the implementation and the concept precision, there is still a lot of different possible choices for the function g abstracting the time precision. There will be many similarities between the six types of protocols, so we start by defining the protocols for the first implementation with infinite precision and for the second implementation with full precision.

To obtain alternative protocols for the first implementation, simply reinterpret the instructions in Figure 4, depending on whether you are considering unbounded precision or fixed precision ϵ. In the first case, we place the masses z_1' and z_2' in the left pans where $z_1' \in (z_1 - 2^{|z_1|}, z_1 + 2^{|z_1|})$ and $z_2' \in (z_2 - 2^{|z_2|}, z_2 + 2^{|z_2|})$; in the second case, we consider $z_1' \in (z_1 - \epsilon, z_1 + \epsilon)$ and $z_2' \in (z_2 - \epsilon, z_2 + \epsilon)$. In this way, we obtain protocols $\textsc{compare}[1, UP]$ and $\textsc{compare}[1, FP(\epsilon)]$. To obtain alternate protocols for the second implementation, simply reinterpret the instructions of the second protocol in Figure 5, depending on whether you are considering unbounded precision or fixed precision ϵ. In the first case, we place the masses z_1' and z_2' in the left pans, where $z_1' \in (z_1 - 2^{|z_1|}, z_1 + 2^{|z_1|})$ and $z_2' \in (z_2 - 2^{|z_2|}, z_2 + 2^{|z_2|})$; in the second case, we consider$z_1' \in (z_1 - \epsilon, z_1 + \epsilon)$ and $z_2' \in (z_2 - \epsilon, z_2 + \epsilon)$. In this way, we obtain protocols $\textsc{compare}[2, UP, g]$ and $\textsc{compare}[2, FP(\epsilon), g]$. Finally, the protocols for the second implementation and different choices of time precision do not differ, since the precision is implicitly present in counting machine transitions.

Reviewing the assumption of uniform probability densities above, we could generalise them to some other computable probability distribution, most obviously the Gaussian distribution. However, the uniform distribution produces simpler arguments, making the exposition clearer, and avoids the problem of estimating the time taken to compute with another distribution.

5. Measuring with the VBE machine with infinite precision

In what follows the suffix operation $\downharpoonright_n$ on a word w, $w\downharpoonright_n$, denotes the prefix sized n of the ω-word $w0^\omega$, no matter the size of w.

Comparing the experimental times relative to two different query words provides information about y. Consider calls with protocol $\textsc{compare}[1, IP]$ for input (z_1, z_2) such that $z_1 < z_2$ and assume that no timeout occurs. The result of the experiment depends only on the relationship between z_1, z_2 and y: (a) if $y < z_1 < z_2$, then the second experiment will end first, (b) if $z_1 < y < z_2$, then any of the two experiments can end first and (c) if $z_1 < z_2 < y$, then the first experiment will end first. We can then deduce the relationship between z_1, z_2 and y.

Proposition 5.1. Let s be the result of $\textsc{compare}[1, IP](z_1, z_2)$, for an unknown mass of size y and time schedule T, such that $|z_1| = |z_2| = n$ and $z_1 < z_2$. Then, (a) if $s =$'FIRST,' then $y > z_1$, (b) if $s =$'SECOND,' then $y < z_2$ and (c) if $s =$'TIMEOUT,' then $|y - z_1| < 2T(|n|)^{-2}$ and $|y - z_2| < 2T(|n|)^{-2}$.

```
ALGORITHM “BINARYSEARCH[$1, IP$]($\ell$)”
“MASS”: Infinite precision case

    input a natural number $\ell$ – number of places to the right of the left leading 0;
    $x_0 := 0$; $x_4 := 1$; $x_2 := (x_0 + x_4)/2$;
    while $x_4 - x_0 > 2^{-\ell}$ do begin
        $x_1 := (x_0 + x_2)/2$;
        $x_3 := (x_2 + x_4)/2$;
        $s_1 :=$ COMPARE[$1, IP$]($x_1{\downarrow_\ell}, x_2{\downarrow_\ell}$);
        $s_2 :=$ COMPARE[$1, IP$]($x_2{\downarrow_\ell}, x_3{\downarrow_\ell}$);
        if $s_1 =$ “SECOND” then $(x_0, x_2, x_4) := (x_0, x_1, x_2)$;
          else if $s_2 =$ “FIRST” then $(x_0, x_2, x_4) := (x_2, x_3, x_4)$;
          else if $s_1 =$ “FIRST” and $s_2 =$ “SECOND” then $(x_0, x_2, x_4) := (x_1, x_2, x_3)$;
          else if $s_1 =$ “TIMEOUT” or $s_2 =$ “TIMEOUT” then $x_0 := x_2$; $x_4 := x_2$
    end while;
    output the dyadic rational denoted by $x_2$.
```

Fig. 6. Procedure to obtain an approximation of an unknown mass of size y placed in the right pan of the balance.

The idea for the measurement is the following: suppose that y lies in the interval $[z_0, z_4]$, where z_0 and z_4 are dyadic rationals. Split the interval in four parts by considering the points z_1, z_2 and z_3, where $z_2 = (z_0 + z_4)/2$, $z_1 = (z_0 + z_2)/2$ and $z_3 = (z_2 + z_4)/2$. Consider protocol calls for the pairs (z_1, z_2) and (z_2, z_3) with experimental times t_1, t_2 and t_3. Depending on the result, we will obtain a new interval where y may belong: (a) if $t_1 > t_2$ (i.e., the first protocol call returns ‘SECOND’), then $y < z_2$ and thus $y \in [z_0, z_2]$, (b) if $t_2 < t_3$ (that is, the second protocol call returns ‘FIRST’), then $y > z_2$ and thus $y \in [z_2, z_4]$, (c) if $t_1 < t_2$ and $t_2 > t_3$ (that is, the first protocol call returns ‘FIRST’ and the second protocol call returns ‘SECOND’), then $z_1 < y < z_3$ and thus $y \in [z_1, z_3]$. Note that the new interval has half the length of the original one and so repeating this process will enable us to obtain approximations to y.

Proposition 5.2. For any unknown mass of size y and any time schedule T, (a) the time complexity of the algorithm of Figure 6, for input ℓ, is $\mathcal{O}(\ell T(\ell))$, (b) for all $k \in \mathbb{N}$, there exists $\ell \in \mathbb{N}$ such that $T(\ell) \geqslant 2^{(k+1)/2}$ and the output relative to the input ℓ is a dyadic rational m such that $|y - m| < 2^{-k}$; moreover, ℓ is at most exponential in k, and (c) if $T(k)$ is exponential in k, then the value of ℓ witnessing $T(\ell) \geqslant 2^{(k+1)/2}$ can be taken to be linear in k.

Proposition 5.3. For any real number $y \in (0, 1)$, there exists a VBE machine $\mathcal{M}(y)$ operating with the type I protocol with infinite precision and exponential schedule is such that: (a) for all size $n \in \mathbb{N}$, $\mathcal{M}(y)$ halts for the input word 1^n and the content of the output tape is a dyadic rational z such that $|z - y| < 2^{-n}$ and (b) the number of steps of $\mathcal{M}(y)$ is bounded by an exponential in n.

Proof: For any $y \in (0,1)$, we take the VBE machine $\mathcal{M}(y)$ with unknown mass y and any exponential time schedule T, operating with type I protocol with infinite precision. According to Proposition 5.2 (b) and (c), there is a constant b such that, for all n, with $\ell = bn$, we have that $T(\ell) \geqslant 2^{(n+1)/2}$. The machine $\mathcal{M}(y)$, on input 1^n, just calls the binary search procedure of Figure 6 with input $\ell = bn$. Since $T(n)$ is exponential in n and ℓ is linear in n, the number of steps of the VBE machine, also in agreement with Proposition 5.2 (a), is in $\mathcal{O}(\ell 2^{a\ell}) \subseteq \mathcal{O}(2^{(a+1)\ell})$, for some $a \in \mathbb{N}$. □

The next step is to provide a measurement algorithm for the type II protocol. We will try to adapt the same algorithm, but now we have to deal with the possibility of getting ‘INDISTINGUISHABLE,’ meaning that both experimental times are too close to separate them. To solve this problem, we compute the time differences as we approach the unknown mass y. It would be interesting if the time differences of a protocol call increase, i.e., if on successive calls to COMPARE$[2, IP, g](z_1, z_2)$, the experimental time differences would become greater. That would mean that the answer ‘INDISTINGUISHABLE’ would not occur (after some point on) and so we could deduce the position of y relative to z_1 and z_2. Fortunately, this is indeed true if both test masses lie on the same side relative to y.

Proposition 5.4. Consider an instance of the VBE with unknown mass of size y and test masses z_1, z_2. Suppose that either $y < z_1, z_2$ or $z_1, z_2 < y$, that $|z_2 - z_1| \geqslant \delta$, $|z_1 - y| \leqslant \zeta$ and $|z_2 - y| \leqslant \zeta$. Then, $|T_{\text{exp}}(z_2, y) - T_{\text{exp}}(z_1, y)| \geqslant a\delta/(\zeta\sqrt{\zeta})$, where $a = y/\sqrt{1+y}$.

Proof: The experimental time and its derivative are given by

$$T_{\text{exp}}(z, y) = \sqrt{\frac{z+y}{|z-y|}} \qquad |T'_{\text{exp}}(z, y)| = \frac{y}{\sqrt{z+y}\sqrt{|z-y|^3}}.$$

To get the desired inequality, we just apply the mean value theorem and the assumption $|z_1 - z_2| \geqslant \delta$. We conclude that there is some value ξ between z_1 and z_2 such that

$$|T_{\text{exp}}(z_2, y) - T_{\text{exp}}(z_1, y)| = |z_2 - z_1||T'_{\text{exp}}(\xi, y)| \geqslant \frac{y\delta}{\sqrt{\xi+y}\sqrt{|\xi-y|^3}} \geqslant \frac{y}{\sqrt{1+y}}\frac{\delta}{\zeta\sqrt{\zeta}},$$

where on the last step we use the facts that $\xi < 1$ and that $|\xi - y| < \zeta$. □

The measurement algorithm for the type II protocol will be very similar to the one for the type I. We begin with the interval $[0,1]$; at step k, we have an interval of size 2^{-k} containing y; then we split the interval in four intervals, obtaining dyadic rationals z_0, z_1, z_2, z_3 and z_4 that are separated by 2^{-k-2}; we perform two protocol calls, one involving z_1 and z_2 and the other involving z_2 and z_3; depending on the answers, we choose one of the intervals (z_0, z_2), (z_1, z_3) or (z_2, z_4) as the next interval, thus obtaining an interval with half of the length of the previous one. To simplify, consider first the case $g = 0$. Suppose that y does not belong to (z_1, z_2). We know that $|z_1 - z_2| = 2^{-k-2}$ and that both $|z_1 - y|$ and $|z_2 - y|$ are at most 2^{-k}. By Proposition 5.4, we obtain that the time difference in protocol call COMPARE$[2, IP, g](z_1, z_2)$ is at least $a \times 2^{k/2}/4$, where a is a constant. In the same way, if y does not belong to (z_2, z_3), the time difference in protocol call COMPARE$[2, IP, g](z_2, z_3)$ is at least $a \times 2^{k/2}/4$. We conclude that the time difference increases exponentially step by step, which implies that we will not get ‘INDISTINGUISHABLE’ whenever the two dyadic rationals lie on the same side relative to y. That is, *if the answer to a protocol call*

COMPARE$[2, IP, g](z, z')$ *with* $z < z'$ *is* 'INDISTINGUISHABLE,' *then* $z < y < z'$. At step k, we make protocol calls with words of size $k + 2$. Suppose that y does not lie in (z_1, z_2), so that the time difference is at least $a \times 2^{k/2}/4$. If $g = 0$, then we get an answer of 'FIRST' (implying that $y > z_1$), 'SECOND' (implying that $y < z_2$) or 'TIMEOUT' (implying that y is very close to z_1 and z_2). We need g to be small enough so that we do not get a wrong answer, that is, the number of machine transitions must not differ too much from the real experimental time. Observe that, for any g, the time differences observed (that is, the difference in the number of machine transitions) is at least $a \times 2^{k/2}/4 - 2g(k + 2)$. Thus, if g is such that $g(k) < a \times 2^{k/2}/16$, then the imprecision in time is not enough to induce a different answer to the protocol call. Thus, any constant time precision, polynomial time precision or exponential time precision $g(n) = c2^{kn}$ with $k < 1/2$ does not prejudice our algorithm. Most of these results are asymptotic, i.e., we can only guarantee that, for a certain point on, the time difference is great enough so that we do not get answers of 'INDISTINGUISHABLE.' However, in the first iterations, it is not impossible to obtain 'INDISTINGUISHABLE' as the answer to COMPARE$[2, IP, g](z, z')$ in a situation where y does not belong in (z, z').

To deal with this problem, note that we could simply begin the measurement with a subinterval of $[0, 1]$, small enough so that it does not happen. The specification of this subinterval only requires a finite amount of information (two dyadic rationals of size k, for k large enough) that only depend on y and g, and thus it could be hard-wired in the measurement algorithm. Thus, we can in fact build a measurement algorithm for the second implementation.

Proposition 5.5. For any unknown mass of size y, any time schedule T and any time precision g such that $g \in o(2^{n/2})$, (a) the time complexity of algorithm of Figure 7 for input ℓ is $\mathcal{O}(\ell T(\ell))$, (b) for all $k \in \mathbb{N}$, there exists $\ell \in \mathbb{N}$ such that $T(\ell) \geqslant 2^{(k+1)/2}$ and the output for input ℓ is a dyadic rational m such that $|y - m| < 2^{-k}$; moreover, ℓ is at most exponential in k, (c) if $T(k)$ is exponential in k, then the value of ℓ witnessing $T(\ell) \geqslant 2^{k/2}$ can be taken to be linear in k.

Proposition 5.6. For any real number $y \in (0, 1)$, there exists a VBE machine $\mathcal{M}(y)$ with the type II protocol with infinite precision in the mass and any time precision $g \in o(2^{n/2})$ such that: (a) for all size $n \in \mathbb{N}$, $\mathcal{M}(y)$ halts for input word 1^n and the content of the output tape is a dyadic rational z such that $|z - y| < 2^{-n}$ and (b) the number of steps of $\mathcal{M}(y)$ is bounded by an exponential in n.

Proof: For any $y \in (0, 1)$, we take the VBE machine $\mathcal{M}(y)$ with the type II protocol with infinite precision and any exponential time schedule T. The machine $\mathcal{M}$, on input 1^n, just calls the binary search procedure of Figure 7 with input $\ell = bn$. The desired approximation is produced with probability 1. Since $T(n)$ is exponential in n and ℓ is linear in n, the number of steps of the VBE machine, in agreement with Proposition 5.5 (a), is in $\mathcal{O}(\ell 2^{an}) \subseteq \mathcal{O}(2^{(a+1)n})$ for some $a \in \mathbb{N}$. □

In the end of this section, we offer a different measurement algorithm for the first implementation. We have seen that it is possible, in polynomial time, to obtain a logarithmic amount of bits of the unknown mass. We will now consider other possibilities

```
ALGORITHM "BINARYSEARCH[$2, IP, g$]"
"MASS": Infinite precision case

    input a natural number $\ell$ — number of places to the right of the left leading 0;
    $x_0 := 0$;
    $x_4 := 1$;
    $x_2 := (x_0 + x_1)/2$;
    while $x_4 - x_0 > 2^{-\ell}$ do begin
        $x_1 := (x_0 + x_2)/2$;
        $x_3 := (x_2 + x_4)/2$;
        $s_1 :=$ COMPARE$[1, IP](x_1\downharpoonright_{\ell+1}, x_2\downharpoonright_{\ell+1})$;
        $s_2 :=$ COMPARE$[1, IP](x_2\downharpoonright_{\ell+1}, x_3\downharpoonright_{\ell+1})$;
        if $s_1 =$ "SECOND" or "INDISTINGUISHABLE" then $(x_0, x_2, x_4) := (x_0, x_1, x_2)$;
          else if $s_2 =$ "FIRST" or "INDISTINGUISHABLE" then $(x_0, x_2, x_4) := (x_2, x_3, x_4)$;
          else if $s_1 =$ "FIRST" and $s_2 =$ "SECOND" then $(x_0, x_2, x_4) := (x_2, x_3, x_4)$;
          else if $s_1 =$ "TIMEOUT" or $s_2 =$ "TIMEOUT" then $x_0 := x_2$; $x_4 := x_2$
    end while;
    output the dyadic rational denoted by $x_2$.
```

Fig. 7. Procedure to obtain an approximation of an unknown mass of size y in the right pan of the balance.

for the measurement. For example, suppose that $y > 1/2$ and we want to measure the point $z > 1/2$ at which $T_{\exp}(z, y) = T_{\exp}(1/2, y)$. With type I protocol, this is indeed possible, as the algorithm of Figure 8 suggests.

Proposition 5.7. Let s be a possible result of COMPARE$[1, IP](1\downharpoonright_\ell, m\downharpoonright_\ell)$ (note that $1\downharpoonright_\ell$ is the ℓ-bits dyadic rational $0.10\cdots0$). For any unknown mass of size $y > 1/2$, let $r > 1/2$ denote the point such that $T_{\exp}(1/2, y) = T_{\exp}(r, y)$. Then, if $s =$'FIRST,' then $r \geqslant m$, and, if $s =$'SECOND,' then $r \leqslant m$.

Proposition 5.8. For any unknown mass of size $1/2 < y < \sqrt{2}/2$ and any time schedule T, let $r > 1/2$ denote the point such that $T_{\exp}(1/2, y) = T_{\exp}(r, y)$. Then, (a) the time complexity of algorithm of Figure 8 for input ℓ is $\mathcal{O}(\ell T(\ell))$ and (b) if $T(\ell) > T_{\exp}(1/2, y)$, then the output for input ℓ is a dyadic rational m such that $|r - m| < 2^{-\ell}$.

Proposition 5.9. For any real number $y \in (1/2, \sqrt{2}/2)$, there exists a VBE machine $\mathcal{M}(y)$ operating on the mass y with type I protocol with infinite precision such that: (a) for all size $n \in \mathbb{N}$, $\mathcal{M}(y)$ halts for the input word 1^n and the content of the output tape is a dyadic rational z such that $|z - r| < 2^{-n}$ and (b) the number of steps of $\mathcal{M}(y)$ is bounded by a polynomial in n.

Proof: The map $z \mapsto \frac{z+1}{2}$ is a bijection $f : [0, 1] \to [1/2, 1]$ that works by simply prefixing a 1 to the binary expansion of the argument z. Suppose we want to measure $r \in (0, 1)$. We take the VBE machine $\mathcal{M}(y)$ with unknown mass y such that $T_{\exp}(1/2, y) = T_{\exp}(f(r), y)$ (it can be easily proved that $r = 2y^2$), and T as any polynomial time schedule. Let ℓ_0 be such that $T(\ell_0) > T_{\exp}(1/2, r)$. The oracle Turing machine $\mathcal{M}(y)$ on input 1^n calls the

```
ALGORITHM "BINARYSEARCH[3, $IP$]"
"MASS": Infinite precision case

    input a natural number $\ell$ – number of places to the right of the left leading 0;
    $x_0 := 1/2$;
    $x_1 := 1$;
    while $x_1 - x_0 > 2^{-\ell}$ do begin
        $m := (x_0 + x_1)/2$;
        $s :=$ COMPARE$[1, IP](1\downharpoonright_\ell, m\downharpoonright_\ell)$;
        if $s =$ "FIRST" then $x_0 = m$ else $x_1 = m$
    end while;
    output the dyadic rational denoted by $m$.
```

Fig. 8. Procedure to obtain an approximation of a real number y, with a test mass z in the right pan of the balance.

procedure BinarySearch$[3, IP](\ell)$, where $\ell = \max\{n+1, \ell_0\}$, from which it gets m, and computes the value $2m - 1$. Since $|f(r) - m| < 2^{-\ell}$, we also have that $|r - (2m-1)| < 2^{-n}$. Since $T(n)$ is polynomial, by Proposition 5.8 (a), the number of steps is bounded by a polynomial. □

6. Measuring with the VBE machine with unbounded precision

The measuring algorithm provided for protocol of type I operating with unbounded precision is very similar to algorithm BinarySearch$[3, IP]$. Its goal is again to measure the value $r > 1/2$ for which $T_{\exp}(1/2, y) = T_{\exp}(r, y)$, where y is the 'unknown mass.'

Lemma 6.1. Let s be a possible result of COMPARE$[1, UP](1\downharpoonright_\ell, m\downharpoonright_\ell)$, for any unknown mass of size $1/2 < y < \sqrt{2}/2$. Let $r > 1/2$ denote the point such that $T_{\exp}(1/2, y) = T_{\exp}(r, y)$. Suppose that ℓ and h are such that $2^{-\ell} < 1/2|1/2 - y|$ and

$$2^h \geqslant \sqrt{(1+y)/(y|y-1/2|^3)} + 1 \ .$$

Then, (a) if $s =$'FIRST,' then $r \geqslant m - 2^{-\ell+h}$ and (b) if $s =$'SECOND,' then $r \leqslant m + 2^{-\ell+h}$.

Proof: When we perform the protocol COMPARE$[1, UP](1\downarrow_\ell, m\downarrow_\ell)$, the test mass to be placed on the first balance, which we denote by z_1, lies in $(1/2 - 2^{-\ell}, 1/2 + 2^{-\ell})$. The imprecision in the mass induces an imprecision in the experimental time, that is, the experimental time $T_{\exp}(z_1, y)$ lies in $(T_{\exp}(1/2, y) - \Delta t, T_{\exp}(1/2, y) + \Delta t)$, for some value of Δt. This imprecision induces an imprecision in the mass close to r, that is, there is an interval $(r - \Delta r, r + \Delta r)$ that contains the values z_2 of masses close to r such that $T_{\exp}(z_2, y) \in (T_{\exp}(1/2, y) - \Delta t, T_{\exp}(1/2, y) + \Delta t)$. The assumption $2^{-\ell} < 1/2|1/2 - y|$ allows us to use the mean value theorem, just as we did in the proof of Proposition 5.4, to estimate Δr. For any $z_1 \in (1/2 - 2^{-\ell}, 1/2 + 2^{-\ell})$, let z_2 be close to r such that $T_{\exp}(z_1, y) = T_{\exp}(z_2, y)$. There are $\xi_1 \in (1/2 - 2^{-\ell}, 1/2 + 2^{-\ell})$ and $\xi_2 \in (r - \Delta r, r + \Delta r)$

```
ALGORITHM "BINARYSEARCH[$1, UP$]($\ell$)"
"MASS": Unbounded precision case

    input a natural number $\ell$ – number of places to the right of the left leading 0;
    $x_0 := 1/2$;
    $x_1 := 1$;
    while $x_1 - x_0 > 2^{-\ell}$ do begin
        $m := (x_0 + x_1)/2$;
        $s :=$ COMPARE[$1, UP$]($1\downharpoonright_\ell, m\downharpoonright_\ell$);
        if $s =$ "FIRST" then $x_0 = m$ else $x_1 = m$
    end while;
    output the dyadic rational denoted by $m$.
```

Fig. 9. Procedure to obtain an approximation of a real number y, with a test mass z in the left pan of the balance.

such that

$$\begin{aligned}|z_1 - \frac{1}{2}||T'_{\exp}(\xi_1, y)| &= |T_{\exp}(z_1, y) - T_{\exp}(1/2, y)| = |T_{\exp}(z_2, y) - T_{\exp}(r, y)| \\ &= |z_2 - r||T'_{\exp}(\xi_2, y)|.\end{aligned}$$

Using the inequalities $\xi_1 \geqslant 0, |\xi_1 - y| \geqslant 1/2|1/2 - y|, \xi_2 \leqslant 1, |\xi_2 - y| \leqslant 1/2$, we obtain, from Proposition 5.4, $|z_2 - r| \leqslant \sqrt{(1+y)/(y|y-1/2|^3)} \times 2^{-\ell}$. Thus, we can take $\Delta r = \sqrt{(1+y)/(y|y-1/2|^3)} \times 2^{-\ell}$. Let us consider again COMPARE[$1, UP$]($1\downarrow_\ell, m\downarrow_\ell$). If $m > r + \Delta r + 2^{-\ell}$, then the result cannot be 'FIRST' (the experimental time $T_{\exp}(m, y)$ is simply too low). By the same reasoning, if $m < r - \Delta r - 2^{-\ell}$, then the result cannot be 'SECOND.' From these two facts, with $2^h \geqslant \sqrt{(1+y)/(y|y-1/2|^3)} + 1$, we obtain the desired results. □

Lemma 6.2. For any unknown mass of size $y > 1/2$ and any time schedule T, let $r > 1/2$ be such that $T_{\exp}(1/2, y) = T_{\exp}(r, y)$ and h be a non-negative integer such that $2^h \geqslant \sqrt{(1+y)/(y|y-1/2|^3)} + 1$. Then, (a) the time complexity of algorithm of Figure 9 for input ℓ is $\mathcal{O}(\ell T(\ell))$ and (b) if $T(\ell) > T_{\exp}(1/2, y)$ and $2^{-\ell} < \sqrt{(1+y)/(y|y-1/2|^3)}$, then the output of algorithm Binary Search 1 UP for input ℓ is a dyadic rational m such that $|r - m| < 2^{-\ell+h+1}$.

Lemma 6.3. For any real number $y \in (0, 1)$, there exists a VBE machine $\mathcal{M}(y)$ operating on the mass y with type I protocol with unbounded precision such that: (a) for all size $n \in \mathbb{N}$, $\mathcal{M}(y)$ halts for the input word 1^n and the content of the output tape is a dyadic rational z such that $|z - r| < 2^{-n}$ and (b) the number of steps of $\mathcal{M}(y)$ is bounded by a polynomial in n.

Proof: This is a consequence of Lemma 6.2. For any real value r, we take $r' = \frac{r+1}{2}$ and VBE as the intended experiment, where the unknown mass y is chosen such that $T_{\exp}(1/2, y) = T_{\exp}(r', y)$. Take T as any polynomial time schedule. Then, there is a constant ℓ_0 such that $T(\ell_0) > T_{\exp}(1/2, y)$ and $2^{-\ell_0} < \sqrt{(1+y)/(y|y-1/2|^3)}$. Take h

```
ALGORITHM "BINARYSEARCH[2, $UP, g$]($\ell$)"
"MASS": Unbounded precision case

    input a natural number $\ell$ — number of places to the right of the left leading 0;
    $x_0 := 0$;
    $x_4 := 1$;
    $x_2 := (x_0 + x_1)/2$;
    while $x_4 - x_0 > 2^{-\ell}$ do begin
        $x_1 := (x_0 + x_2)/2$;
        $x_3 := (x_2 + x_4)/2$;
        $s_1 :=$ COMPARE[2, $UP, g$]($x_1\lfloor_{\ell+3}, x_2\lfloor_{\ell+3}$);
        $s_2 :=$ COMPARE[2, $UP, g$]($x_2\lfloor_{\ell+3}, x_3\lfloor_{\ell+3}$);
        if $s_1 =$ "SECOND" or "INDISTINGUISHABLE" then $(x_0, x_2, x_4) := (x_0, x_1, x_2)$;
          else if $s_2 =$ "FIRST" or "INDISTINGUISHABLE" then $(x_0, x_2, x_4) := (x_2, x_3, x_4)$;
          else if $s_1 =$ "FIRST" and $s_2 =$ "SECOND" then $(x_0, x_2, x_4) := (x_2, x_3, x_4)$;
          else if $s_1 =$ "TIMEOUT" or $s_2 =$ "TIMEOUT" then $x_0 := x_2$; $x_4 := x_2$
    end while;
    output the dyadic rational denoted by $x_2$.
```

Fig. 10. Procedure to obtain an approximation of an unknown mass y in the right pan of the balance.

such that $2^h \geqslant \sqrt{(1+y)/(y|y-1/2|^3)} + 1$. The oracle Turing machine, for input 1^k, consists in a call to BinarySearch[1, UP](ℓ), where $\ell = \max\{k + h + 2, \ell_0\}$, obtaining a result m, and then returns the value $2m - 1$. Since $|r' - m| < 2^{-\ell+h+1}$, we also have that $|r - (2m-1)| < 2^{-k}$ with probability 1. Since $T(k)$ is polynomial, the number of steps is then bounded by a polynomial. □

The measuring algorithm for protocol type 2 operating with unbounded precision and tolerance g is different from the measurement algorithm for protocol type 1 operating with unbounded precision. When we were considering protocol type 1 operating with infinite precision, we saw that, in COMPARE[2, IP, g](z_1, z_2) at step k, we had $|z_1 - z_2| = 2^{-k-2}$ and $|z_1 - a|, |z_2 - a| \leqslant 2^{-k}$. In the following algorithm, we make oracle queries for words of size $\ell + 3$. In this way, we guarantee that at step k, $k = 0, \ldots \leqslant \ell - 1$, the imprecision in placing the test masses is at most $2^{-\ell-3} \leqslant 2^{-k-4}$, thus ensuring that in the call to COMPARE[2, UP, g](z_1, z_2), we have that $|z_1 - z_2| \geqslant 2^{-k-3}$. In this way, we obtain a time difference greater than or equal to *const.* × $2^{k/2}/8$. Therefore, our algorithm will work for all choices of time precision g such that $g \in o(2^{n/2})$.

Lemma 6.4. For any unknown mass of size y, any time schedule T and any time precision g such that $g \in o(\lambda n.\ 2^{n/2})$, let s be a possible result of one of the calls COMPARE[2, UP, g]($z_1\lfloor_{\ell+3}, z_2\lfloor_{\ell+3}$) during the execution of algorithm of Figure 10. Then, (a) if $s =$ 'FIRST,' then $y \geqslant z_1 - 2^{-\ell-3}$, (b) if $s =$ 'SECOND,' then $y \leqslant z_2 + 2^{-\ell-3}$, (c) if $s =$ 'INDISTINGUISHABLE,' then $z_1 - 2^{-\ell-3} \leqslant y \leqslant z_2 + 2^{-\ell-3}$ and (d) if $s =$ 'TIMEOUT,' then $|y - z_1| < (\mu/T(\ell+3))^2 + 2^{-\ell-3}$ and $|y - z_2| < (\mu/T(\ell+3))^2 + 2^{-\ell-3}$.

Proposition 6.5. For any unknown mass of size y, any time schedule T and any time precision g such that $g \in o(\lambda n.\ 2^{n/2})$, (a) the time complexity of algorithm of Figure 10 on input ℓ is $\mathcal{O}(\ell T(\ell))$, (b) for all $k \in \mathbb{N}$, there exists $\ell \in \mathbb{N}$ such that $T(\ell+3) \geqslant \mu 2^{(k+1)/2}$ and thus the output of the same algorithm for input ℓ is a dyadic rational m such that $|y-m| < 2^{-k}$; moreover, ℓ is at most exponential in k and (c) if $T(k)$ is exponential in k, then the value of ℓ witnessing $T(\ell) \geqslant \mu 2^{k/2}$ can be taken to be linear in k.

Proposition 6.6. For any real number $y \in (0,1)$, there exists a VBE machine $\mathcal{M}(y)$ operating on the mass y with type II protocol with unbounded precision and time tolerance $g \in o(\lambda n.\ 2^{n/2})$ such that: (a) for all size $n \in \mathbb{N}$, $\mathcal{M}(y)$ halts for the input word 1^n and the content of the output tape is a dyadic rational z such that $|z-r| < 2^{-n}$ and (b) the number of steps of $\mathcal{M}(y)$ is bounded by a polynomial in n.

Proof: This is a consequence of Proposition 6.5. For any real value y, we take VBE as the intended experiment with unknown mass y. Take T as any exponential time schedule. The oracle Turing machine, for input 1^n, consists simply in a call to BinarySearch$[1, IP](\ell)$, where ℓ is chosen such that $\ell \geqslant n+1$ and $T(\ell+3) \geqslant 2^{(n+2)/2}$ (moreover, ℓ is linear in n), thus producing the desired approximation with probability 1. Since $T(n)$ is exponential in n and ℓ is linear in n, the number of steps of the VBE machine is bounded by a function in $\mathcal{O}(\ell 2^{an}) \subseteq \mathcal{O}(2^{(a+1)n})$, for some a. □

7. Measuring with the VBE machine with fixed precision

The final case to consider is that of fixed precision. The measurement algorithms that we are going to specify do not produce approximations to a given mass. Instead, the goal is to compute the approximation to the probability of some particular event. We are going to obtain approximations to the probability of getting 'FIRST' from COMPARE$[\ldots](10\downarrow_\ell, 11\downarrow_\ell)$, corresponding to the test masses 1/2 and 3/4, with fixed precision ϵ for the mass and time tolerance g, either in the type I or type II protocols.

Proposition 7.1. Let T be a time schedule such that $T(2) > T_{\exp}(1/2, 3/4) = \sqrt{5}$. For any mass $y \in (1/2, 3/4)$, let $P_{\text{FIRST}}(y)$ denote the probability of obtaining result 'FIRST' in protocol COMPARE$[1, FP(\epsilon)](10, 11)$ (resp. COMPARE$[2, FP(\epsilon), g](10, 11)$, for any time tolerance g) for the experiment with test mass y. Then, for any sufficiently small ϵ, $P_{\text{FIRST}}(1/2) = 0$, $P_{\text{FIRST}}(3/4) = 1$ and P_{FIRST} is a continuous, increasing function in $(1/2, 3/4)$.

The above proposition entails that, by the intermediate value theorem, for any intended probability p, there is some mass y such that $P_{\text{FIRST}}(y) = p$, and so we can compute approximations to p by setting y as desired and repeating several times the protocol call COMPARE$[\ldots](10, 11)$. All that is left is to formalise the algorithm and state its properties.

Proposition 7.2. In the fixed precision scenario, with protocol type I or II, with any time tolerance, for any time schedule T such that $T(2) > T_{\exp}(1/2, 3/4)$, there exists a sufficiently small ϵ such that, for any unknown mass of size y and natural number h, (a) the time complexity of algorithm FreqCountFP(ϵ, h) of Figure 11 is $\mathcal{O}(2^{2\ell})$, where ℓ is

```
ALGORITHM "FREQCOUNTFP($\epsilon, h$)"
"MASS": Finite precision case

    input a natural number $\ell$ – used to set the precision of the approximation;
    $counter := 0$;
    $\zeta := 2^{2\ell+h-2}$;
    repeat $\zeta$ times
        $s :=$ COMPARE[$\mathcal{P}$](10, 11);  %P is both for the first and the second types
        if $s =$ "FIRST" then $counter := counter + 1$
    end repeat;
    output the dyadic rational denoted by $counter/\zeta$.
```

Fig. 11. Procedure to obtain an approximation of a real probability p, assuming fixed precision ϵ and unknown mass of size y such that $P_{\text{FIRST}}(y) = p$. The value h is an integer number used to bound the probability of error.

the input, and (b) with probability of error 2^{-h}, the output of algorithm FreqCountFP(ϵ, h) on input ℓ is a dyadic rational m such that $|P_{\text{FIRST}}(y) - m| < 2^{-\ell}$.

Proof: The procedure COMPARE[$\mathcal{P}$](10, 11) can be seen as a Bernoulli trial with probability of success p. Let α be the quantity of experiments returning 'FIRST' in $\zeta = 2^{2\ell+h-2}$ trials and X denote the estimator $X = \alpha/\zeta$, we conclude that $\mathbb{E}[X] = p$ and $\mathbb{V}[X] \leqslant 1/(4\zeta)$. The probability that $|X - p| > 2^{-\ell}$ can be bounded by the Chebyshev's inequality, $P(|X - p| > 1/2^{\ell}) \leqslant \mathbb{V}[X] \times 2^{2\ell} \leqslant 2^{2\ell}/(4\zeta) = 2^{-h}$. □

Proposition 7.3. For any real number $y \in (0, 1)$, for all sufficiently small ϵ, and for all $\gamma \in (0, 1/2)$, there exists a VBE machine $\mathcal{M}(y)$ clocked in exponential time operating, either with type I protocol or type II protocol with arbitrary tolerance, with fixed precision ϵ, such that, for every $n \in \mathbb{N}$, every computation of $\mathcal{M}(y)$ on input word 1^n halts with a dyadic rational z as output, such that, with probability of failure at most γ, $|z - r| < 2^{-n}$.

Proof: This is a consequence of Proposition 7.2. We take any time schedule T such that $T(2) > T_{\exp}(1/2, 3/4)$, any $\epsilon \in (0, 1/2)$ in the conditions of Proposition 7.1 and, for any $\gamma \in (0, 1)$, any positive integer number h such that $2^{-h} \leqslant \gamma$. Now, consider the oracle Turing machine that, for input word 1^{ℓ}, makes a call to FreqCountFP(ϵ, h)(ℓ). For any real number r, we take the VBE machine with unknown mass y chosen such that $P_{\text{FIRST}}(y) = r$, where P_{FIRST} is given by Proposition 7.1. The above machine produces the desired approximation with probability of failure at most $2^{-h} \leqslant \gamma$. Furthermore, the number of steps is bounded by a function of the order of $\mathcal{O}(2^{2\ell})$. □

8. Tossing coins with the VBE machine

The final step before establishing lower bounds is to find out which protocols can be used with the VBE to simulate coin tosses. As expected, any probabilistic protocol suffices; that is, only Type I and Type II protocols operating with infinite precision and infinite precision and tolerance 0, respectively, do not allow for fair coin tosses.

Proposition 8.1. The VBE machine operating with type I protocol with unbounded or fixed precision for sufficiently small ϵ can simulate arbitrarily long sequences of fair independent coin tosses, to within a specified chance of failure.

Proof: For a given y, let z be a dyadic rational such that $z \neq y$ and let $\ell \geqslant |z|$ be such that $T(\ell) > T_{\exp}(z,y)$. Observe that both calls $\text{COMPARE}[\ldots, UP, \ldots](z\downharpoonright_\ell, z\downharpoonright_\ell)$ and $\text{COMPARE}[\ldots, FP(\epsilon), \ldots](z\downharpoonright_\ell, z\downharpoonright_\ell)$ have more than one possible result (in fact, both return 'FIRST' or 'SECOND' with the same, non-null probability). □

Proposition 8.2. The VBE machine operating with type II protocol with infinite, unbounded, or fixed precision, for sufficiently small ϵ and non-null tolerance g, permits coin tosses.

Proof: For a given y, let z be a dyadic rational such that $z \neq y$ and let $\ell \geqslant |z|$ be such that $T(\ell) > T_{\exp}(z,y)$ and $g(\ell) > 0$. Now, observe that protocol call $\text{COMPARE}[2, P, g](z\downharpoonright_\ell, z\downharpoonright_\ell)$, where P is any of the protocol variants, has more than one possible result (in fact, both return 'FIRST' or 'SECOND' with the same, non-null probability). □

Proposition 8.3. The VBE machine operating with type II protocol with unbounded or fixed precision, for sufficiently small ϵ and tolerance 0, permits coin tosses.

Proof: For a given unknown mass y, we will find a dyadic rational z of size ℓ such that the protocol call $\text{COMPARE}[2, UP, 0](z\downharpoonright_\ell, z\downharpoonright_\ell)$ (resp. $\text{COMPARE}[2, FP(\epsilon), 0](z\downharpoonright_\ell, z\downharpoonright_\ell)$) returns 'FIRST' or 'SECOND' with the same probability. We will require that $T(\ell) > T_{\exp}(z,y)$ (this ensures that we do not get 'TIMEOUT' with probability 1) and $\lceil T_{\exp}(z - 2^{-\ell}, y)\rceil \neq \lceil T_{\exp}(z + 2^{-\ell}, y)\rceil$ (resp. $\lceil T_{\exp}(z - \epsilon, y)\rceil \neq \lceil T_{\exp}(z + \epsilon, y)\rceil$; this ensures that we do not get 'INDISTINGUISHABLE' with probability 1). The above constraints are satisfied by considering a large enough integer k such that equation $T_{\exp}(x,y) = k$, for fixed y, has a solution x. We then take ℓ such that $T(\ell) > k$ and z of size ℓ such that $z - 2^{-\ell} < x \leqslant z$ (resp. $z - \epsilon < x \leqslant z$). □

9. Encoding discrete functions as real numbers

Denote the *Cantor numbers* by $\mathcal{C}_3$, the set of real numbers x such that $x = \sum_{k=1}^{\infty} x_k 2^{-3k}$, where $x_k \in \{1, 2, 4\}$, i.e., the numbers composed by triples of the form 001, 010 or 100.

Proposition 9.1. For every $x \in \mathcal{C}_3$ and for every dyadic rational $z \in (0,1)$ with size $|z| = m$, (a) if $|x - z| \leqslant 1/2^{i+5}$, then the binary expansions of x and z coincide in the first i bits and (b) $|x - z| > 1/2^{m+10}$.

Proof: (a) First, suppose that z and x coincide on the first $i - 1$ bits and differ on the ith bit. We have two relevant cases.

$z < x$: In this case, $z_i = 0$ and $x_i = 1$. In the worst cases, the binary expansion for z after the ith position begins with a sequence of 1s and the binary expansion for x after the ith position begins with a sequence of 0s:

	i	lower bound of $\lvert x - z \rvert$
z	$\cdots 011111 \cdots$	
x (case $i \equiv_3 0$)	$\cdots 100100 \cdots$	$> 2^{-(i+3)}$
x (case $i \equiv_3 1$)	$\cdots 100001 \cdots$	$> 2^{-(i+5)}$
x (case $i \equiv_3 2$)	$\cdots 100010 \cdots$	$> 2^{-(i+4)}$

$z > x$: In this case, $z_i = 1$ and $x_i = 0$. In the worst cases, the binary expansion for z after the ith position begins with a sequence of 0s and the binary expansion for x after the ith position begins with a sequence of 1s:

	i	lower bound of $\lvert x - z \rvert$
z	$\cdots 1000 \cdots$	
x (case $i \equiv_3 0$)	$\cdots 0100 \cdots$	$> 2^{-(i+2)}$
x (case $i \equiv_3 1$)	$\cdots 0101 \cdots$	$> 2^{-(i+2)}$
x (case $i \equiv_3 2$)	$\cdots 0110 \cdots$	$> 2^{-(i+3)}$

We conclude that in any case $|x - z| > 2^{-(i+5)}$. Thus, if $|x - z| \leqslant 2^{-(i+5)}$, then x and z coincide in the first i bits.

(b) Since the binary expansion of z after the mth bit is exclusively composed of 0s and any Cantor number $x \in \mathcal{C}_3$ has at most four consecutive 0s in its binary expansion, we conclude that, in the best fit, z and x can not coincide in the $m + 5$th bit. Thus, by (a), $|x - z| > 2^{-(m+10)}$. □

Now, we will encode a given advice function $f : \mathbb{N} \to \{0, 1\}^\star$ into a real number in $(0, 1)$, using # as a string delimiter.

Definition 9.2. The encoding of a word $w \in \Sigma^\star$, denoted by $c(w)$, is the binary expression of the real number obtained first by converting w to a string of 0's and 1's, and then replacing every 0 by 100 and every 1 by 010. Given a function $f \in \log\star$, we denote the encoding of f by the real number $\mu(f) = \lim \mu(f)(n)$, recursively defined by (a) $\mu(f)(0) = 0 \cdot c(f(0))$, (b) $\mu(f)(n + 1) = \mu(f)(n)\#c(s)$, whenever $f(n + 1) = f(n)\#s$ and $n + 1$ is not a power of two and (c) $\mu(f)(n + 1) = \mu(f)(n)\#c(s)\#001$, whenever $f(n + 1) = f(n)\#s$ and $n + 1$ is a power of two.

The encoding above consists on replacing the bits of f by triples 100 and 010, adding 001 at the end of each code of $f(2^k)$, with $k \in \mathbb{N}$. Observe that, by construction, $\mu(f) \in \mathcal{C}_3$. Also, from the encoding, we need to get back values of the advice f. To obtain $f(2^k)$, we just have to read the bits of $\mu(f)$ in triples until the $(k + 1)$-th triple 001 is found. Consequently, one may say that, whenever $f \in \log\star$, by knowing $\mathcal{O}(k)$ bits of the real number $\mu(f)$, we can know $f(2^k)$.

Bounds similar to those in the proposition above have other uses – compare the role of Proposition 3.5 in the continuity studies in Pauly and Ziegler (2013).

10. Lower bounds on the computational power of the VBE machine

We studied a variety of protocols sufficient enough to prove the following theorems.

Proposition 10.1. If $A \in P/poly$, then A is decidable in polynomial time by a VBE machine $\mathcal{M}(y)$ for some suitable y operating with type I protocol using infinite precision.

Proof: This proof of this proposition follows the steps of Proposition 10.3, mutatis mutandis. □

Proposition 10.2. (a) If $A \in P/poly$, then A is decidable in polynomial time by a VBE machine $\mathcal{M}(y)$ for some suitable y operating with type I protocol using unbounded precision. (b) If $A \in BPP//log\star$, then A is decidable in polynomial time by a VBE machine operating with type I protocol and sufficiently small fixed precision ϵ.

Proof: This proof of this proposition follows the steps of Proposition 10.4, mutatis mutandis, noting that $P/poly = BPP//poly$. □

Proposition 10.3. If $A \in P/log\star$, then A is decidable in polynomial time by a VBE machine $\mathcal{M}(y)$ for some suitable y operating with type II protocol using infinite precision and tolerance zero.

Proof: Let f be a prefix function in log and $\mathcal{M}$ be a Turing machine running in polynomial time such that, for every $n \in \mathbb{N}$ and every word w of size less than or equal to n, $w \in A$ if and only if $\mathcal{M}$ accepts $\langle w, f(n)\rangle$. Take y to be the encoding of f as a real number in $(0, 1)$. According to Proposition 5.6, there exists an integer k, an oracle Turing machine $\mathcal{M}'(y)$ and a time schedule T, such that (a) for every $n \in \mathbb{N}$, $\mathcal{M}'(y)$ on input 1^n halts and outputs the dyadic rational z such that $|z - \gamma| < 2^{-n}$ and (b) the number of steps of $\mathcal{M}'(y)$ is bounded by a function in $2^{\mathcal{O}(n)}$. Our next step is to define the oracle Turing machine $\mathcal{M}''(y)$ that decides A. For a given input w of size n, we perform a sequence of experiments to compute $f(n')$, for some $n' > n$. We take $n' = 2^{\lceil \log(n) \rceil}$. In this way, we can obtain $f(n')$, if we know the binary expansion of y up to the $(\lceil \log(n) \rceil + 1)$-th triple of the form 001.

Since $f \in$ log, this means that there are constants a and b such that, for all n, $|f(n)| \leqslant a\log(n) + b$. In particular, $|f(n')| \leqslant a\lceil \log(n) \rceil + b$. Thus, we need to know at most the first $3(a\lceil \log(n) \rceil + b) + 3(\lceil \log(n) \rceil + 1)$ bits of the binary expansion of y to get to the desired triple 001. Finally, we specify our VBE machine, $\mathcal{M}''(y)$ using time schedule T. For a given input word w of size n, it simulates the machine $\mathcal{M}'(y)$ for input 1^ℓ, where $\ell = 3(a+1)\lceil \log(n) \rceil + 3b + 8$. By the discussion above, the result is a dyadic rational z such that $|z - \gamma| < 2^{-\ell}$ and thus z and γ coincide in the first $\ell - 5 = 3(a+1)\lceil \log(n) \rceil + 3(b+1)$ bits, and so z can be used to decode $f(n')$. Afterwards, simulate $\mathcal{M}$ for the input word $\langle w, f(n')\rangle$ and accept or reject based on the result of the simulation. It is clear that this machine decides A. The time complexity of the simulation of $\mathcal{M}'(y)$ is $\mathcal{O}(2^{a\ell})$. Since ℓ is logarithmic on n, the result is polynomial in n. And since $\mathcal{M}$ runs in polynomial time, we conclude that $\mathcal{M}''(y)$ also runs in polynomial time. □

Proposition 10.4. (a) If $A \in BPP//log\star$, then A is decidable in polynomial time by a VBE machine operating with type II protocol with unbounded precision and time tolerance

$g \in o(2^{n/2})$. (b) If $A \in BPP//log\star$, then A is decidable in polynomial time by a VBE machine operating with type II protocol with sufficiently small fixed precision and any time tolerance.

Proof: We prove for the case of unbounded precision. The proof relative to the fixed precision case is similar. We use a proof similar to the proof of Proposition 10.3, but now we have to take into account both the error probability in measuring and the simulation of a fair coin toss.

Let f be a prefix function in log, γ_1 be a real number in $(0, 1/2)$ and $\mathcal{N}$ be a Turing machine running in polynomial time such that, for any natural number n and any word w of size less than or equal to n,

$$\text{if } w \in A, \text{ then } \mathcal{N} \text{ rejects } \langle w, f(n)\rangle \text{ with probability at most } \gamma_1;$$
$$\text{if } w \notin A, \text{ then } \mathcal{N} \text{ accepts } \langle w, f(n)\rangle \text{ with probability at most } \gamma_1.$$

Let p_3 be a polynomial bound on the running time of $\mathcal{N}$. Let also r be the coding $y = \mu(f)$ and γ_3 be such that $\gamma_1 + \gamma_3 < 1/2$. Then, according with Propositions 6.6 (unbounded precision) and 7.3 (fixed precision), there is an integer k, an oracle Turing machine $\mathcal{M}(y)$, and a time schedule T such that: (a) for all size n, every computation of $\mathcal{M}(y)$, for input word 1^n, halts and, with probability of failure at most γ_3, the content of the output tape is a dyadic rational m such that $|m - y| < 2^{-n}$ and (b) the number of steps of $\mathcal{M}(y)$ is bounded by a function in $\mathcal{O}(2^{an})$.

Furthermore, in agreement with Propositions 8.2 or 8.3, we conclude that there is a dyadic rational z such that the protocol call with query z has more than one possible result with non-null probability. This means that protocol call has a probability of δ of producing some result r_1, with $\delta \in (0, 1)$. Let also γ_2 be a positive real number such that $\gamma_1 + \gamma_2 + \gamma_3 < 1/2$. There is an integer K (depending on δ and γ_2) such that, for all n, we can use Kn independent biased coin tosses to simulate n fair coin tosses, with probability of failure at most γ_2.

Our next goal is to define the oracle Turing machine $\mathcal{M}'$ that will be used to decide A. Figure 12 synthesises the description of the behavior of the intended oracle Turing machine. Given input w of size n, the idea is again to use the experiment to compute $f(n')$, where $n' = 2^{\lceil \log n \rceil}$. For that purpose, at most the first $3(C_1\lceil\log(n)\rceil + C_2) + 3(\lceil\log(n)\rceil + 1)$ bits of the binary expansion of y are needed, for some constants C_1 and C_2. Thus, begin by simulating $\mathcal{M}$ for input word 1^ℓ where $\ell = 3(C_1 + 1)\lceil\log(n)\rceil + 3C_2 + 8$. By the above discussion, with probability of failure at most γ_3, the result is a dyadic rational m such that $|m - y| < 2^{-\ell}$ and thus, by Proposition 9.1, m and γ coincide in the first $\ell - 5 = 3(C_1 + 1)\lceil\log(n)\rceil + 3(C_2 + 1)$ bits, and so m can be used to decode $f(n')$.

In the next step, use the dyadic rational z to produce a sequence of $Kp_3(n)$ independent biased coin tosses, and then attempt to extract from that sequence $p_3(n)$ fair coin tosses. In case of failure (with probability at most γ_2), simply reject the input word. Otherwise, simulate $\mathcal{N}$ for the input word $\langle w, f(n')\rangle$, using the sequence of fair coin tosses to choose the path of computation. To finish the computation, accept or reject based on the result of the simulation.

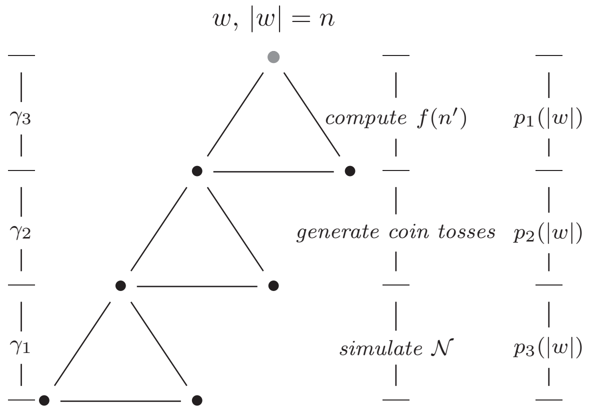


Fig. 12. Behaviour of the oracle Turing machine $\mathcal{M}'$.

Let us see that the machine decides A. If $w \in A$, then the machine may reject w if the wrong approximation of $f(n')$ was produced or if it failed in producing the sequence of independent coin tosses or if the simulation of $\mathcal{N}$ rejected $\langle w, f(n')\rangle$. This happens with probability at most $\gamma_1 + \gamma_2 + \gamma_3$. If $w \notin A$, then the machine may accept A if the wrong approximation of $f(n')$ was produced or if the simulation of N accepted $\langle w, f(n')\rangle$, which happens with probability at most $\gamma_1 + \gamma_3$. So this means that the probability of failure is bounded by $\gamma_1 + \gamma_2 + \gamma_3 < 1/2$.

Let us see that the machine runs in polynomial time. The time complexity of the first step is $\mathcal{O}(2^{a\ell})$, which is again bounded by some polynomial in n, $p_1(n)$. The second step also ends in some polynomial time $p_2(n)$ since we need only $Kp_3(n)$ biased coin tosses, which is a polynomial amount, and each coin toss takes constant time. And since $\mathcal{N}$ runs in polynomial time p_3, we conclude that $\mathcal{M}'$ runs in polynomial time $p_1 + p_2 + p_3$. □

11. Upper bounds on the computational power of the VBE machine

Given a vanishing value oracle (that is, an oracle with three or four possible random answers), we can depict the sequence of the answers in a binary tree, where each path is labelled with its probability. The leaves of these trees are marked with ACCEPT or REJECT. Then, to get the probability of acceptance of a particular word, we simply add the probabilities for each path that ends in acceptance. The next basic idea is to think of what would happen if we change the probabilities in the tree. This means that we are using the same procedure of the Turing machine, but now with a different probabilistic oracle. Suppose that the tree has depth m and there is a real number β that bounds the difference in the probabilities labelling all pairs of corresponding edges in the two trees. Proposition 2.1 of Beggs et al. (2009), states that the difference in the probabilities of acceptance of the two trees is at most $2m\beta$. We need to state and prove a result equivalent to this one, but for 3-adic and 4-adic trees (see Figure 13). In Beggs et al. (2009), we defined $f_d(m, \beta)$ as the largest possible difference in probabilities of acceptance for two different assignments of probabilities with difference at most β in a d-adic probabilistic tree of depth m.

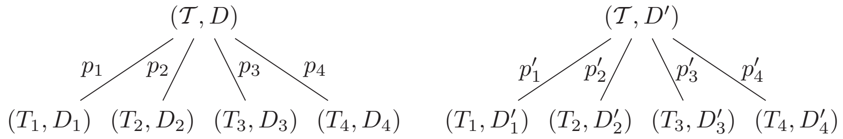


Fig. 13. Proving Proposition 11.2.

Definition 11.1. Let d be an integer with $d \geqslant 2$. By a d-adic probabilistic tree, we mean a pair $(\mathcal{T}, D)$, where:

— $\mathcal{T}$ is a tree with some set of nodes or vertices V, some set of edges E and some set of leaves $L \subseteq V$;

— $D : E \to [0, 1]$ is a map that assigns to each edge u a probability $D(u)$;

— $\mathcal{T}$ is a d-adic tree, that is, each inner node has exactly d children; moreover, if $u_1, \ldots, u_d$ are its outgoing edges, then $D(u_1) + \cdots + D(u_d) = 1$;

— Each leaf is either an accepting node (labelled with ‘A’) or a rejecting node (labelled with ‘R’).

Given two d-adic probabilistic trees $(\mathcal{T}, D)$ and $(\mathcal{T}, D')$ with the same d-adic tree $\mathcal{T}$ (see Figure 13), we define the distance $d(D, D')$, as $d(D, D') = \max_{u \in E} \ |D(u) - D'(u)|$. Let $\mathcal{T}^d_m$ denote the set of d-adic trees of depth at most m and $\mathcal{T} \in \mathcal{T}^d_m$. Define $f_d : \mathbb{N} \times [0, 1] \to [0, 1]$ by

$$f_d(m, \beta) = \max_{d(D,D') \leqslant \beta} |P(\mathcal{T}, D) - P(\mathcal{T}, D')|.$$

Thus, $f_d(m, \beta)$ gives the largest possible difference in probabilities of acceptance for two different assignments of probabilities with difference at most β.

Proposition 11.2. For any $m \in \mathbb{N}$ and $\beta \in [0, 1]$, $f_3(m, \beta) \leqslant 2m\beta$ and $f_4(m, \beta) \leqslant 3m\beta$.

Proof: This proposition is a generalisation of Proposition 2.1 in Beggs et al. (2009). The proof involves some algebra but is more or less straightforward. □

Proposition 11.3. If A is decidable in polynomial time by the VBE machine $\mathcal{M}(y)$ for $y \in (0, 1)$ operating with any protocol $\mathcal{P}$ with exponential time schedule, then $A \in P/poly$.

Proof: Let A be a set decidable by the VBE machine operating with protocol $\mathcal{P}$ with an exponential schedule T. Let c and d be constants such that, for an input word of size n, all queries have size at most $\lceil c \log n + d \rceil$. A query of size k is defined by two dyadic rationals of size k, so there are exactly 2^{2k} possible queries of size k. This means that the number of different possible queries in a computation with an input word of size n is at most polynomial in n:

$$\sum_{i=1}^{\lceil c \log n + d \rceil} 2^{2i} < \frac{2^{2d+2}}{3} n^{2c} \ .$$

If $\mathcal{P}$ is deterministic, then the advice function f is the concatenation of all triples $\langle z_1, z_2, r \rangle$ such that z_1 and z_2 are dyadic rationals of size $k \leqslant \lceil c \log n + d \rceil$ and r encodes the result of protocol call COMPARE$[\mathcal{P}](z_1, z_2)$. (There are either three or four possible results, so r may be encoded with only two bits.) If $\mathcal{P}$ is probabilistic, then the advice function f is the concatenation of all tuples $\langle z_1, z_2, p_f, p_s, p_t, p_u\rangle$ such that z_1 and z_2 are dyadic rationals of size $k \leqslant \lceil c\log n + d\rceil$ and p_f, p_s, p_t, p_u are approximations to the probabilities of obtaining each possible result ('FIRST,' 'SECOND,' 'TIMEOUT' or 'INDISTINGUISHABLE') of protocol call COMPARE$[\mathcal{P}](z_1, z_2)$. We will approximate each probability with a dyadic rational of size k such that the error in the probability of any query is bounded by 2^{-k}. To find the suitable value of k, we observe that the VBE machine deciding A runs in polynomial time $\mathcal{O}(n^a)$ and so the number of possible queries in the computation of any word of size n is at most bn^a, for some constant b. The probabilistic tree induced by the computation has a depth at most bn^a. Now, if γ is the bound on the error probability associated with the machine, by Proposition 11.2, we take k such that $3bn^a2^{-k} < 1/2 - \gamma$, that is, $2^k > 3bn^a/(1/2 - \gamma)$. Such a k can then be taken to be logarithmic in n. In either case, $f \in$ poly since f is the concatenation of a polynomial amount of tuples each with logarithmic size.

We specify a Turing machine $\mathcal{M}$ to decide A in polynomial time using f as advice. $\mathcal{M}$ simulates the VBE machine for the same input word. When reaching a query state with query words z_1 and z_2, it reads the appropriate tuple in f. In the deterministic case, the machine resumes the computation in the proper outcome state. In the probabilistic case, the machine uses the approximations to the probabilities of each result and choose one of them with k coin tosses. In the deterministic case, it is clear that this machine decides A in polynomial time. In the probabilistic case, this machine induces a probabilistic tree in the same way as the original machine, with depth less than bn^a and edge difference lower than 2^{-k}. Then, by Proposition 11.2 and the above calculations, the difference in the probabilities of acceptance is then bounded by a constant less than $1/2 - \gamma$. Thus, the probability that this machine gives a wrong answer is bounded by a constant less than $1/2$, and so the machine specified decides A in polynomial time. □

This proposition is not enough for our purposes: in the first implementation, we will be looking for non-exponential time schedules and in the second implementation, we want to establish the upper bounds of $P/log\star$ or $BPP//log\star$.

11.1. *Upper bounds for type I protocol.*

The technique that will be applied to type I protocol relies on Proposition 11.5, that can be used to decide the result of an oracle query for the infinite precision case. Recalling timing for the VBE in 2.2, we begin by defining the boundary numbers.

Definition 11.4. Let $y \in (0, 1)$ be the size of the unknown mass and T the time schedule of a particular VBE machine. Then, for every natural number k, we define ℓ_k and r_k as the real numbers in $(0, 1)$ such that $\ell_k < y < r_k$ and $T_{\exp}(\ell_k, y) = T_{\exp}(r_k, y) = T(k)$.

Proposition 11.5. Let $y \in (0, 1)$ be the size of the unknown mass and T the time schedule of a particular VBE machine for the first protocol type operating with infinite precision. Let z_1 and z_2 be two dyadic rationals of size k. Let s be the result of the protocol call COMPARE$[1, IP](z_1, z_2)$. Then, (a) if $\ell_k \leqslant z_1, z_2 \leqslant r_k$, then $s =$'TIMEOUT,' (b) if $z_1 < \ell_k \leqslant z_2 < r_k$ or $\ell_k \leqslant z_2 \leqslant r_k < z_1$, then $s =$'FIRST,' (c) if $z_2 < \ell_k \leqslant z_1 \leqslant r_k$ or $\ell_k \leqslant z_1 \leqslant r_k < z_2$, then $s =$'SECOND,' (d) if $z_1 < z_2 < \ell_k$ or $r_k < z_2 < z_1$, then $s =$'FIRST,' (e) if $z_2 < z_1 < \ell_k$ or $r_k < z_1 < z_2$, then $s =$'SECOND,' (f) if $z_1 < \ell_k < r_k < z_2$ and $z_1z_2 \leqslant y^2$, then $s =$'FIRST,' (g) if $z_1 < \ell_k < r_k < z_2$ and $z_1z_2 > y^2$, then $s =$'SECOND,' (h) if

```
ALGORITHM "SIMULATE$(1, IP)(z_1, z_2)$"

input two dyadic rational numbers $z_1$ and $z_2$ with same size $k$;
Advice consists of three dyadic rational numbers $\ell_k$, $r_k$ (with size $k$) and $y^2$ (with size $2k$);

    if $\ell_k \leq z_1, z_2 \leq r_k$ then return "TIMEOUT";
    if $z_1 < \ell_k \leq z_2 \leq r_k$ or $\ell_k \leq z_2 \leq r_k < z_1$, then return "FIRST";
    if $z_2 < \ell_k \leq z_1 \leq r_k$ or $\ell_k \leq z_1 \leq r_k < z_2$, then return "SECOND";
    if $z_1 < z_2 < \ell_k$ or $r_k < z_2 \leq z_1$, then return "FIRST";
    if $z_2 < z_1 < \ell_k$ or $r_k < z_1 \leq z_2$, then return "SECOND";
    if $z_1 < \ell_k < r_k < z_2$ and $z_1 z_2 \leq y^2$, then return "FIRST";
    if $z_1 < \ell_k < r_k < z_2$ and $z_1 z_2 > y^2$, then return "SECOND";
    if $z_2 < \ell_k < r_k < z_1$ and $z_1 z_2 > y^2$, then return "FIRST";
    if $z_2 < \ell_k < r_k < z_1$ and $z_1 z_2 \leq y^2$, then return "SECOND".
```

Fig. 14. Procedure to simulate an oracle query of size k; it receives as advice the suitable approximations of the boundary numbers ℓ_k and r_k and of y^2, where y is the size of the unknown mass.

$z_2 < \ell_k < r_k < z_1$ and $z_1 z_2 > y^2$, then $s =$'FIRST' and (i) if $z_2 < \ell_k < r_k < z_1$ and $z_1 z_2 \leqslant y^2$, then $s =$'SECOND.'

Proof: All the cases except for the last four are obvious. For the last four cases, we can assume, without loss of generality, that $z_1 < \ell_k < r_k < z_2$. To know the answer of protocol call COMPARE$[1, IP](z_1, z_2)$, we need to compare $T_{\exp}(z_1, y)$ and $T_{\exp}(z_2, y)$. Using the fact that $z_1 < y$ and $z_2 > y$, a quick calculation reveals that $T_{\exp}(z_1, y) < T_{\exp}(z_2, y)$, if and only if $\sqrt{(z_1 + y)/(y - z_1)} < \sqrt{(z_2 + y)/(z_2 - y)}$, if and only if $z_1 z_2 < y^2$. □

We can then use the algorithm of Figure 14 to simulate oracle queries for the first oracle type with infinite precision.

Proposition 11.6. If A is a set decidable in polynomial time by the VBE machine with type I protocol operating with infinite precision, then $A \in P/poly$. (Note the difference relative to Proposition 11.3: Proposition 11.6 is independent of the time schedule.)

Proof: Let $\mathcal{M}(y)$ be the VBE machine with protocol of the first type operating with infinite precision that decides A in polynomial time. Since $\mathcal{M}(y)$ runs in polynomial time, there is a polynomial bound bn^a to the size of the queries during the computation relative to an input of size n. Consider the advice function f such that

$$f(n) = \ell_1\downharpoonleft_1 \# r_1\downharpoonleft_1 \# \cdots \# \ell_t\downharpoonleft_t \# r_t\downharpoonleft_t \# y^2\downharpoonleft_{2t} \ ,$$

where y is the unknown mass associated with $\mathcal{M}$, ℓ_k and r_k are the corresponding boundary numbers and $t = bn^a$, so that $f \in$ poly. Now, consider the Turing machine $\mathcal{M}'$ with advice f that, for an input word of size n, simulates the VBE machine $\mathcal{M}(y)$ and, whenever in the query state, runs the algorithm SIMULATE$(1, IP)$ for the input z_1 and z_2 of size k, i.e., the content of the query tape, using $\ell_k\downharpoonleft_k$, $r_k\downharpoonleft_k$ and $y^2\downharpoonleft_{2k}$ as advice. Clearly, $\mathcal{M}'$ has the same behaviour as $\mathcal{M}(y)$ and, since each simulation can be done in polynomial time, this machine also runs in polynomial time. It follows that $A \in P/poly$. □

For the unbounded precision case, remember that when a word z is written on the query tape the actual test mass is a real number uniformly and independently sampled in

```
ALGORITHM "SIMULATE[1, $UP$]($z_1, z_2$)"
Input two dyadic rational numbers $z_1$, $z_2$ of size $k$ and the desired precision $s$;

  Advice consists of three dyadic rational numbers $\ell_k$, $r_k$ (with size $k+s$)
          and $y^2$ (with size $2(k+s)$);
  Randomly choose a dyadic rational $z_1'$ of size $k+s$ in $(z_1 - 2^{-k}, z_1 + 2^{-k})$;
          %This can be done with $s+1$ coin tosses
  Randomly choose a dyadic rational $z_2'$ of size $k+s$ in $(z_2 - 2^{-k}, z_2 + 2^{-k})$;
          %This can be done with $s+1$ coin tosses
  Return the output of SIMULATE[1, $IP$]($z_1', z_2'$) with advice ($\ell_k, r_k, y^2$).
```

Fig. 15. Procedure to simulate an oracle query of size k; it receives as advices approximations of the boundary numbers ℓ_k and r_k and of y^2, where y is the unknown mass.

the interval $(z - 2^{-|z|}, z + 2^{-|z|})$. To simulate an oracle query, we will first randomly choose a dyadic rational of size $|z| + s$ in the interval $(z - 2^{-|z|}, z + 2^{-|z|})$ (that is, we approach the real test mass with a dyadic test mass); this can be done with $s+1$ coin tosses. Doing this twice, we obtain two dyadic rationals z_1' and z_2' close to z_1 and z_2, respectively; then we simulate an experiment with infinite precision with z_1' and z_2', using approximations of ℓ_k, r_k and y^2 (using the algorithm SIMULATE[1, IP]). The whole idea is described in Figure 15.

We observe that the algorithm SIMULATE[1, UP](z_1, z_2, s) is probabilistic, like the protocol call COMPARE[1, UP](z_1, z_2). The next step should be to bound the difference in probabilities between these two procedures. As the following proposition shows, this bound depends on the *precision s of the quantiser*.

Proposition 11.7. If p is the probability of obtaining result 'FIRST,' 'SECOND' or 'TIMEOUT' in the protocol call COMPARE[1, UP](z_1, z_2), for the unbounded precision case, for an unknown mass y and any time schedule T, and q is the probability of obtaining the same result in algorithm SIMULATE[1, UP](z_1, z_2, s), receiving as advice approximations of y^2 as well as the boundary numbers ℓ_k and r_k associated with y and T, then $|p - q| < 2^{-s+1}$.

Proof: When protocol call COMPARE[1, UP](z_1, z_2) is made for dyadic rationals of size k, we can think of the actual test masses used as a point (ξ, υ) uniformly and independently sampled in the two-dimensional region $R = (z_1 - 2^{-k}, z_1 + 2^{-k}) \times (z_2 - 2^{-k}, z_2 + 2^{-k})$. This region can be divided in three different regions: (a) the region R_f, where the result is 'FIRST,' that is defined by the equations

$$(\xi < \ell_k \wedge \xi < \upsilon < y^2/\xi) \vee (\xi > r_k \wedge y^2/\xi < \upsilon < \xi)\ ,$$

(b) the region R_s, where the result is 'SECOND,' that is defined by the equations

$$(\upsilon < \ell_k \wedge \upsilon < \xi < y^2/\upsilon) \vee (\upsilon > r_k \wedge y^2/\upsilon < \xi < \upsilon)\ ,$$

(c) the region R_t, where the result is 'TIMEOUT,' that is defined by the equations

$$(\ell_k < \xi < r_k) \wedge (\ell_k < \upsilon < r_k).$$

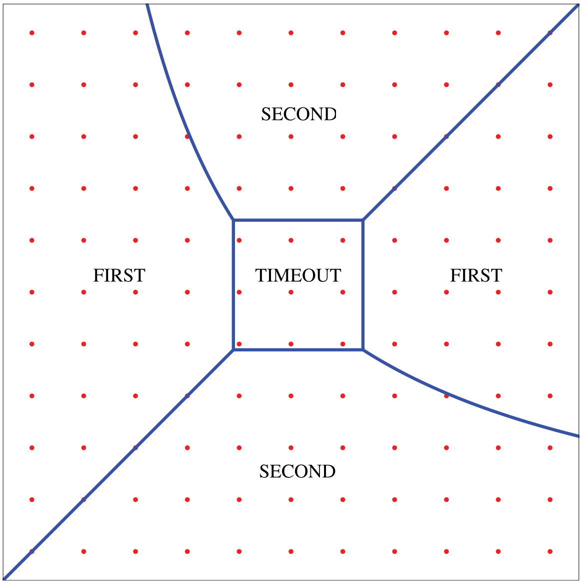


Fig. 16. Regions.

Then, the probability of obtaining some result ‘FIRST,’ ‘SECOND’ or ‘TIMEOUT’ is simply the area of the corresponding region divided by the area of the full square, which is 2^{-2k+2}. Figure 16 shows the various regions for a possible situation.

When we are simulating the oracle query with algorithm SIMULATE$[1, UP]$, we are basically approximating each region by a union of small squares; we are dividing R into an array of 2^{s+1} by 2^{s+1} squares, each of these squares has a representative (ξ, υ) where ξ and υ are dyadic rationals of size 2^{k+s}. Then, the probability of obtaining a given result is simply the number of squares for which its representative falls in the corresponding region, divided by the total number of squares, which is 2^{2s+2}. To bound the difference in probability, we observe that this difference comes from the squares containing points in more than one region. We call these squares *tainted*. Observe that, whenever a square is not tainted, that is, completely confined to one of the regions R_f, R_s, R_t, it does not contribute to the error in the probability used in the simulation. In other words, *only the tainted squares contribute to the error*. This step is decisive, since it implies that the absolute difference $|p - q|$ is bounded by the total area of the tainted squares. Finally, a simple counting argument reveals that the total number of tainted squares is at most $2(2 \times 2^{s+1} - 1) - 1 < 2^{s+3}$. In this way, we obtain the desired bound in the difference of probabilities, as $|p - q| < 2^{s+3}/2^{2s+2} = 2^{-s+1}$. □

Proposition 11.8. If A is a set decidable in polynomial time by the VBE machine operating with type I protocol and unbounded precision, then $A \in P/poly$.[¶]

Proof: Let A be decidable in polynomial time by the VBE machine operating with type I protocol and unbounded precision. We will prove that $A \in BPP//poly$. There is a

[¶] Again, this result does not depend on any particular time schedule.

polynomial bound bn^a on both the size of a query and the number of queries that can be made during the computation on an input word of size n. We want to approximate the probability of any possible result of any possible query with a dyadic rational of size e, for a suitable e, such that the difference in probabilities of any query is bounded by 2^{-e}. If γ is the bound on the error probability associated with the VBE machine, then, by Proposition 11.2, the suitable value of e must be such that $2 \times bn^a 2^{-e} < 1/2 - \gamma$, that is, $2^e > 2bn^a/(1/2 - \gamma)$. (The probabilistic tree induced by the VBE machine is ternary in this case.) Such a e can be taken to be logarithmic in n.

Now, we consider the advice function f such that

$$f(n) = \ell_1\downharpoonleft_{2+e}\#r_1\downharpoonleft_{2+e}\#\cdots\#\ell_t\downharpoonleft_{t+e+1}\#r_t\downharpoonleft_{t+e+1}\#y^2\downharpoonleft_{2(t+e+1)} ,$$

where y is the unknown mass associated with the VBE machine, ℓ_k and r_k are the corresponding boundary numbers and $t = bn^a$. It is immediate that $f \in$ poly. We specify a machine for deciding the set A in polynomial time, using f as advice. This machine simulates the VBE machine for the same input word. Whenever in a query state with query words z_1 and z_2 of size k, it runs algorithm $\textsc{simulate}(1, UP)$ for the input z_1, z_2 (in the query tape) and $e + 1$, using $\ell_k\downharpoonleft_{k+e+1}$, $r_k\downharpoonleft_{k+e+1}$ and $y^2\downharpoonleft_{2(k+e+1)}$ as advice. This machine induces a probabilistic tree in the same way as the original machine, with depth lower than bn^a. Thanks to Proposition 11.7, the edge difference is also lower than 2^{-e}. Then, by Proposition 11.2, the difference in the probabilities of acceptance is bounded by a constant less than $1/2 - \gamma$. Since each simulation can be done in polynomial time, this machine also runs in polynomial time. It follows that $A \in BPP//poly = P/poly$. □

The remaining case is that of the fixed precision ϵ. The idea is the same as in the unbounded precision case, that is, we will devise a randomised algorithm to simulate oracle queries. For two given dyadic rationals z_1, z_2, we discretise the region

$$R = (z_1 - \epsilon, z_1 + \epsilon) \times (z_2 - \epsilon, z_2 + \epsilon) .$$

In the unbounded precision case, it was easy to select a dyadic rational of fixed size in the interval $(z - 2^{-|z|}, z + 2^{-|z|})$ since the amplitude was itself a dyadic rational; for the fixed precision, we have to deal with the fact that ϵ may not be a dyadic rational. The idea that we will use is as follows: let t be a natural number such that $2^{-t-1} < \epsilon \leqslant 2^{-t}$; we randomly choose a dyadic rational $z' \in (z - 2^{-t}, z + 2^{-t})$ of size σ (this can be done with $\sigma - t + 1$ coin tosses);[‖] then with probability at least 1/2, we obtain that $z' \in (z - \epsilon, z + \epsilon)$. If we repeat the above procedure h times, then we can get a dyadic rational of size σ with probability of failure less than 2^{-h}. Moreover, all dyadic rationals of size σ in the interval have the same probability of being chosen. This is the core idea for simulating oracle queries with fixed precision.

Proposition 11.9. If p is the probability of obtaining result '$\textsc{first}$,' '$\textsc{second}$' or '$\textsc{timeout}$' in the protocol call $\textsc{compare}[1, FP(\epsilon)](z_1, z_2)$, for an unknown mass y and any time schedule T, and q is the probability of obtaining the same result in algorithm

[‖] Observe that σ may be smaller than $|z|$; i.e., the dyadic rational generated has smaller size than z; the intuitive meaning of this is that increasing the size of the queries does not contribute to increase their precision.

```
Algorithm "simulate$(1, FP(\epsilon))$"

input two dyadic rational numbers $z_1$ and $z_2$ with size $k$, the desired precision $\sigma$, and $h$;
Advice consists of four dyadic rational numbers $\ell_k$, $r_k$ (size $\sigma$), $y^2$ (size $2\sigma$) and $\epsilon$ (size $\sigma$);
Find $t$ such that $2^{-t-1} < \epsilon \leq 2^{-t}$; %Just count the number of 0s in the head of $\epsilon$
repeat $h$ times
    Randomly choose a dyadic rational $z_1'$ of size $\sigma$ in $(z_1 - 2^{-t}, z_1 + 2^{-t})$;
            %This can be done with $\sigma - t + 1$ coin tosses
    if $z_1' \in (z_1 - \epsilon, z_1 + \epsilon)$ then break;
end repeat;
if $z_1' \notin (z_1 - \epsilon, z_1 + \epsilon)$ then return "timeout";
repeat $h$ times
    Randomly choose a dyadic rational $z_2'$ of size $\sigma$ in $(z_2 - 2^{-t}, z_2 + 2^{-t})$;
            %This can be done with $\sigma - t + 1$ coin tosses
    if $z_2' \in (z_2 - \epsilon, z_2 + \epsilon)$ then break;
end repeat;
if $z_2' \notin (z_2 - \epsilon, z_2 + \epsilon)$ then return "timeout";
simulate$[1, IP](z_1', z_2')$ with advice $(\ell_k, r_k, y^2)$
```

Fig. 17. Procedure to simulate an oracle query of size k; it receives as advice approximations of the boundary numbers ℓ_k and r_k, of y^2, where y is the unknown mass, and of the fixed precision ϵ. Observe that the result 'timeout' in two instructions is irrelevant; we could choose any other result, since the probability of the algorithm ending in any of this instructions will decrease to 0 as we increase the value of h.

simulate$[1, FP(\epsilon)](z_1, z_2, \sigma, h)$ of Figure 17 receiving as advice approximations of y^2 and ϵ as well as the boundary numbers ℓ_k and r_k associated with y and T, then $|p - q| < 2^{-h+1} + \epsilon\, 2^{-\sigma+3} + 2^{-\sigma+2}/\epsilon$.

Proof: There are three situations that change the probability of a given result: (a) the algorithm fails to produce the desired dyadic rational of size σ, (b) the algorithm does not take into account a small area in the outer part of the region R, and (c) the algorithm has different probabilities in the inner part of the region R. The first situation occurs with probability less than $2^{-h} + 2^{-h}$. Regarding the second situation, let $N = \lfloor \epsilon \times 2^{\sigma} \rfloor$; observe that the dyadic rationals produced by the algorithm that are in $(z - \epsilon, z + \epsilon)$ belong in fact to the interval $(z - N2^{-\sigma}, z + N2^{-\sigma})$; this means that we are approaching region R of Figure 16 by region $R' = (z_1 - N2^{-\sigma}, z_1 + N2^{-\sigma}) \times (z_2 - N2^{-\sigma}, z_2 + N2^{-\sigma})$. This means that we must account for the difference between the areas of these two regions, which is bounded by $4 \times 2\epsilon \times 2^{-\sigma}$. Finally, the calculations for the third situation are the same as in the proof of Proposition 11.7; we divide R' by an array of $2N$ by $2N$ squares; counting the number of tainted squares, we conclude that the difference in probabilities is less than $2/N$. Now, using the fact that $N > \epsilon 2^{\sigma}/2$, we obtain the desired bound of $2^{-h+1} + \epsilon\, 2^{-\sigma+3} + 2^{-\sigma+2}/\epsilon$. □

Proposition 11.10. If A is a set decidable in polynomial time by the VBE machine operating with type I protocol and fixed precision ϵ, then $A \in P/poly$.

Proof: Let A be decidable in polynomial time by the VBE machine operating on a mass y with type I protocol and fixed precision ϵ. First, we observe that, to obtain a bound of 2^{-e} on the difference of probability in any oracle query, it suffices to consider h and σ such that $h = e+2$ and $\sigma = e+3+\lceil\log(2\epsilon+1/\epsilon)\rceil$ and invoke Proposition 11.9. (The constant $3+\lceil\log(2\epsilon+1/\epsilon)\rceil$ can be hard-wired into the machine.) Since there is a polynomial bound bn^a on both the size of a query and the number of queries that can be made during the computation on an input word of size n, we take e such that $2^e > 2bn^a/(1/2-\gamma)$. Next, we consider the advice function f such that

$$f(n) = \ell_1\downharpoonleft_\sigma\#r_1\downharpoonleft_\sigma\#\cdots\#\ell_t\downharpoonleft_\sigma\#r_t\downharpoonleft_\sigma\#y^2\downharpoonleft_{2\sigma}\#\epsilon\downharpoonleft_\sigma,$$

where y is the unknown mass associated with VBE machine, ℓ_k and r_k are the corresponding boundary numbers and $t = bn^a$. It is immediate that $f \in$ poly (actually, σ is logarithmic in n).

The machine that decides the set A in polynomial time, using f as advice, simply simulates the VBE machine for the same input word and replaces oracle calls by algorithm SIMULATE$[2, FP(\epsilon)]$ using as input the content of the query tape as well as k and σ, and using as advice the appropriate ℓ_k and r_k as well as y^2 and ϵ. This machine induces a probabilistic tree in the same way as the original machine. By Proposition 11.2 and the above discussion, the difference in the probabilities of acceptance is bounded by a constant less than $1/2-\gamma$. Since each simulation can be done in polynomial time, this machine also runs in polynomial time. It follows that $A \in BPP//poly = P/poly$. □

Proposition 11.11. If A is a set decidable in polynomial time by the VBE machine operating with type I protocol and fixed precision ϵ, then $A \in BPP//log\star$.

Proof: We strengthen the proof of Proposition 11.10 assuming without loss of generality that, as a function of n, $T(n)^2 > 2$. The boundary numbers ℓ_k and r_k are such that

$$\ell_k = y\left(1-\frac{2}{T(k)^2+1}\right), \qquad r_k = y\left(1+\frac{2}{T(k)^2-1}\right).$$

We can use $d+2$ digits of y to obtain a precision of d digits on ℓ_k and r_k. As in the previous proof, σ can be taken as logarithmic in n, i.e., $\sigma = \xi\lceil\log(n)\rceil + e$, with $\xi, e \in \mathbb{N}$. We define an auxiliary function $\tilde{f}$ such that (a) $\tilde{f}(0)$ is the concatenation of the first $e+2$ bits of y, the first $2e$ bits of y^2 and the first e bits of ϵ, and (b) $\tilde{f}(t+1)$ is the concatenation of $\tilde{f}(t)$, the bits from $e+\xi t+3$ to $e+\xi t+\xi+2$ of y, the bits from $2e+2\xi t+1$ to $2e+2\xi t+\xi$ of y^2 and the bits from $e+\xi t+1$ to $e+\xi t+\xi$ of ϵ.

We can use $\tilde{f}(t)$ to obtain the first $\xi t+e+2$ digits of y, the first $2\xi t+2e$ digits of y^2, and the first $\xi t+e$ digits of ϵ. By the previous discussion, we can then use the approximations of y to compute the first $\xi t+e$ bits of any real ℓ_i or r_i. Also, we can see that $|\tilde{f}(t)| = \xi t+e+2+2\xi t+2e+\xi t+e = \mathcal{O}(t)$. Finally, the advice function required is $\tilde{g}(n) = \tilde{f}(\lceil\log(n)\rceil)$. Observe that $\tilde{g}$ is a prefix function and that $|\tilde{g}(n)| = \mathcal{O}(\log(n))$. Furthermore, $\tilde{g}(n)$ can be used to compute $f(n)$ on the proof of Proposition 11.10. Now, we specify a probabilistic machine for deciding set A in polynomial time, using $\tilde{g}$ as advice. Simply retrieve f from $\tilde{g}$ and then use the machine specified in the proof of the previous proposition. Since this retrieval can be made in polynomial time and the above

```
Algorithm "random"
input natural numbers $g$ – defines the range $[-g, g]$ – and $h$ – number of iterations;
$s = \lceil \log(2g+1) \rceil$;  % 2g+1 is the number of possible results
repeat $h$ times
        Randomly choose a natural number $r \in [0, 2^s)$;  %It can be done with s coin tosses;
        if $r \leq 2g$ then break;  %This happens with probability greater than 1/2
end repeat;
if $r > 2g$ then return "FAIL";
return $r - g$.
```

Fig. 18. Probabilistic procedure to find an integer in the range $[-g, g]$ with probability of failure bounded by 2^{-h}.

machine also runs in polynomial time, we conclude that A is decided by this procedure in polynomial time, as we wanted to prove. □

11.2. *Upper bounds for type II protocol.*

We consider now the type II protocol. In this implementation, there are two aspects to consider in order to simulate oracle queries: (a) the need to compute the exact number of machine steps that an experiment takes and (b) the need to produce integer numbers in the range $[-g(k), g(k)]$, where g is the time tolerance and k is the query size. The first is solved with a suitable advice and the second is solved by the randomiser of Figure 18.

Proposition 11.12. For any $h \in \mathbb{N}$, (a) the time complexity of algorithm RANDOM for an input g of size n and number h is $\mathcal{O}(hn)$ and (b) with probability of failure less than 2^{-h}, the output of algorithm RANDOM for input g is an integer in $[-g, g]$.

We will use algorithm RANDOM several times in the following proofs; observe also that not all time tolerances are simulatable; in particular, we prove upper bounds under the assumption that g is computable in polynomial time. We also need to define a sequence of real numbers, similar to the boundary numbers.

Definition 11.13. Let $y \in (0, 1)$ be the size of the unknown mass of a particular VBE machine. We define the section numbers ℓ'_k and r'_k for every $k \in \mathbb{N}$, as the real numbers in $(0, 1)$ such that $\ell'_k < y < r'_k$ and $T_{\exp}(\ell'_k, y) = T_{\exp}(r'_k, y) = k$.

The section numbers effectively split the interval $[0, 1]$ according to the corresponding experimental time. Moreover, given a dyadic rational z, in order to compute the time taken for an experiment with test mass z (assuming infinite precision and zero time tolerance), we simply need to find k such that either $\ell'_{k-1} < z \leqslant \ell'_k$ or $r'_k < z \leqslant r'_{k-1}$. Observe also that the boundary numbers are a particular case of section numbers since $\ell_k = \ell'_{T(k)}$ and $r_k = r'_{T(k)}$.

Proposition 11.14. If A be a set decidable in polynomial time by the VBE machine operating with type II protocol with infinite precision and time tolerance $g \in PF$, then $A \in P/poly$.

```
ALGORITHM "SIMULATE$[2, IP, g]$"
input two dyadic rational numbers $z_1$ and $z_2$ – both with same size $k$ –
        and a natural number $h$ – precision desired;
Advice consists of a sequence of dyadic rationals $\ell''_1, \dots, \ell''_T, r''_T, \dots, r''_1$
        approximating the section numbers;
$T$ is half of the number of dyadic rationals in the above sequence;
Find $T_1$ such that $\ell''_{T_1-1} < z \le \ell''_{T_1}$ or $r''_{T_1} < z \le r''_{T_1-1}$;
        if no such $T_1$ exists then $T_1 = T + 1$;
Find $T_2$ such that $\ell''_{T_2-1} < z \le \ell''_{T_2}$ or $r''_{T_2} < z \le r''_{T_2-1}$;
        if no such $T_2$ exists then $T_2 = T + 1$;
$T_1 := T_1 + \text{RANDOM}(g(k))$;
$T_2 := T_2 + \text{RANDOM}(g(k))$;
Compare $T_1$ and $T_2$:
if $T_1 > T$ and $T_2 > T$ then return "TIMEOUT";
if $T_1 < T_2$ then return "FIRST";
if $T_1 > T_2$ then return "SECOND";
if $T_1 = T_2$ then return "INDISTINGUISHABLE".
```

Fig. 19. Procedure to simulate an oracle query of size k; it receives as advice approximations $\ell''_1, \dots, \ell''_T$ and $r''_1, \dots, r''_T$ of the section numbers $\ell'_1, \dots, \ell'_T$ and $r'_1, \dots, r'_T$ as in the proof of Proposition 11.14; assume that $g \in PF$.

Proof: Let $\mathcal{M}(y)$ be the VBE machine operating with mass y and deciding A in polynomial time with protocol SIMULATE$(2, IP, g)$ of Figure 19, where $g \in PF$. In the case that $g \not\equiv 0$, let $\gamma \in (0, 1/2)$ bound the probability of failure. Since $\mathcal{M}(y)$ runs in polynomial time, there is a polynomial bn^a such that, for every input word of size n: (a) it bounds the number of possible queries and (b) it bounds the maximum possible size of a query. Consider the advice function f such that $f(n) = \ell'_1\downharpoonleft_t \# \ell'_2\downharpoonleft_t \# \cdots \# \ell'_t\downharpoonleft_t \#\# r'_t\downharpoonleft_t \# \cdots \# r'_2\downharpoonleft_t \# r'_1\downharpoonleft_t$, where $t = bn^a$. That is, $f(n)$ is a non-decreasing sequence of dyadic rationals of size t that divide the interval $[0, 1]$ in intersecting sub-intervals corresponding to each experimental time. Observe that timeouts occur when the test mass lies in (ℓ'_t, r'_t).

The machine that decides the set A in polynomial time, using f as advice, simply simulates $\mathcal{M}$ for the same input word of size n. Whenever in a query state, $\mathcal{M}(y)$ sequentially compares the query words z_1 and z_2 with the dyadic rationals in the advice, thus obtaining T_1 and T_2 such that $\lceil T_{\exp}(z_1, y)\rceil = T_1$ and $\lceil T_{\exp}(z_2, y)\rceil = T_2$. In the case that $\ell_t < z_i < r_t$ (that is, the experiment times out), we take T_i as $t + 1$.

In the case that g is the null function, we can simply resume the computation in the corresponding state ('FIRST,' 'SECOND,' 'TIMEOUT' or 'INDISTINGUISHABLE') by comparing T_1, T_2 and t. For a non-null g, we produce two random integers r_1 and r_2 with two calls to RANDOM$(g(|z_1|), h)$ for a suitable value of h.[††] If any of these calls fail (with probability less than 2^{-h}), we resume the computation in the state 'TIMEOUT.' Otherwise, we compare $T_1 + r_1, T_2 + r_2$ and t and resume the computation in the corresponding state. To find the suitable value of h, observe that this machine induces a probabilistic tree in the same way

[††] This is the step in which we assume that $g \in PF$.

```
ALGORITHM "SIMULATE[$2, UP, g$]"

input two dyadic rational numbers $z_1$ and $z_2$ — both with same size $k$
    — and natural numbers $s$ and $h$ — desired precision;
Advice consists of a sequence of dyadic rationals $\ell''_1, \dots, \ell''_T, r''_T, \dots, r''_1$
        approximating the section numbers;
Randomly choose a dyadic rational $z'_1$ of size $k+s$ in $(z_1 - 2^{-k}, z_1 + 2^{-k})$;
        %This can be done with $s+1$ coin tosses
Randomly choose a dyadic rational $z'_2$ of size $k+s$ in $(z_2 - 2^{-k}, z_2 + 2^{-k})$;
        %This can be done with $s+1$ coin tosses
return the output of SIMULATE[$2, IP, g$]$(z'_1, z'_2, h)$ with the same advice.
```

Fig. 20. Procedure to simulate an oracle query of size k; it receives as advice approximations $\ell''_1, \dots, \ell''_t$ and $r''_1, \dots, r''_t$ of the section numbers $\ell'_1, \dots, \ell'_t$ and $r'_1, \dots, r'_t$ as in the proof of the Proposition 11.15; assume that $g \in PF$.

as $\mathcal{M}(y)$, with edge difference lower than 2×2^{-h}. The depth of the tree is bounded by t which is polynomial in n. Thus, we take h such that $3 \times t \times 2 \times 2^{-h} < 1/2 - \gamma$, that is, $2^h > 6t/(1/2-\gamma)$, so that h is polylogarithmic in n. Then, by Proposition 11.2, the difference in the probabilities of acceptance is bounded by a constant less than $1/2 - \gamma$ and so the probability that this machine gives a wrong answer is bounded by a constant less than $1/2$. In either case ($g \equiv 0$ or $g \not\equiv 0$), each simulation can be done in polynomial time, so this machine also runs in polynomial time. It follows that $A \in P/poly$ or $A \in BPP//poly$, which are the same. □

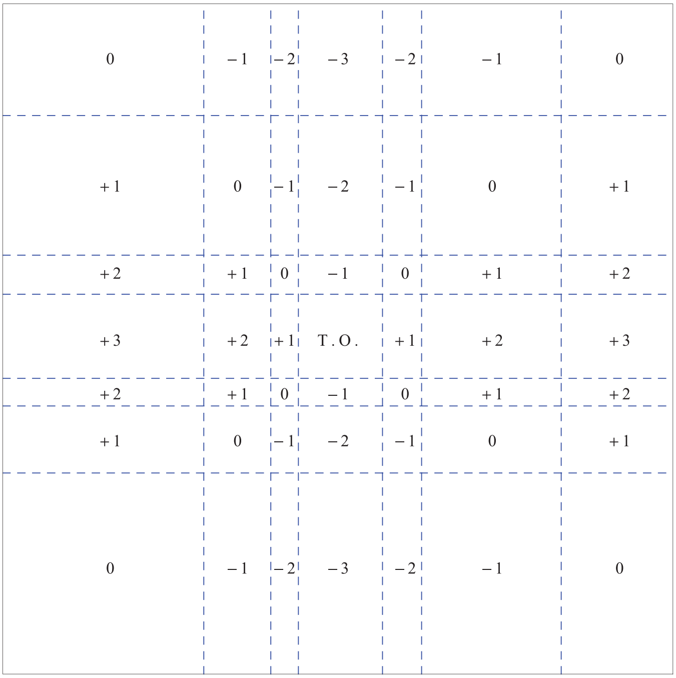
For the other precision cases, we again consider a technique similar to quantisation. To simulate an oracle query with test masses z_1 and z_2 of size k, we first generate in a random way dyadic rationals z'_1 and z'_2 of a suitable size close to z_1 and z_2, respectively; then we simulate an experiment with infinite precision with z'_1 and z'_2 using algorithm SIMULATE$[2, IP, g]$.

Proposition 11.15. If p be the probability of obtaining either result 'FIRST,' 'SECOND,' 'TIMEOUT' or 'INDISTINGUISHABLE' in protocol COMPARE$[2, UP, g](z_1, z_2)$, for an unknown mass y, any time schedule T, and any time tolerance $g \in PF$, and q is the probability of obtaining the same result in algorithm SIMULATE$[2, UP, g](z_1, z_2, s, h)$ of Figure 20, where z_1 and z_2 have size k, receiving as advice dyadic rationals $\ell''_1, \dots, \ell''_t, r''_t, \dots, r''_1$ approximating the section numbers $\ell'_1, \dots, \ell'_t, r'_t, \dots, r'_1$ such that $|\ell''_i - \ell'_i|$, $|r''_i - r'_i| < 2^{-k-s}$, then $|p - q| < 2^{-h+1} + 2^{-s+2}\, T(k)$.

Proof: We can think that the actual test masses used are points (ξ, υ) uniformly and independently sampled in the two-dimensional region:

$$R = (z_1 - 2^{-k}, z_1 + 2^{-k}) \times (z_2 - 2^{-k}, z_2 + 2^{-k}) \; .$$

This region can be divided in regions R_i, $i \in \mathbb{N}$, where R_i is the set of points (ξ, υ) such that $\lceil T_{\exp}(\xi, y) \rceil - \lceil T_{\exp}(\upsilon, y) \rceil = i$. Then, the probability of obtaining a time difference of i (not accounting for the time tolerance) is simply the area of the corresponding region



0	−1	−2	−3	−2	−1	0
+1	0	−1	−2	−1	0	+1
+2	+1	0	−1	0	+1	+2
+3	+2	+1	T.O.	+1	+2	+3
+2	+1	0	−1	0	+1	+2
+1	0	−1	−2	−1	0	+1
0	−1	−2	−3	−2	−1	0

Fig. 21. Example of the regions, for the square $[0,1] \times [0,1]$, with $y = 1/2$.

divided by the area of the full square, which is 2^{-2k+2}. Figure 21 shows the various regions for a possible situation.

Finally, the probability of obtaining a given result 'FIRST,' 'SECOND,' 'TIMEOUT' or 'INDISTINGUISHABLE' is the weighted sum of the above probabilities, such that region R_i has a weighted probability corresponding to the probability of obtaining that result given that the time difference is i. When we are simulating the oracle query with algorithm SIMULATE$[2, UP, g]$, we are dividing R into an array of 2^{s+1} by 2^{s+1} squares, each of these squares has a representative (ξ, υ), where ξ and υ are dyadic rationals of size 2^{k+s}. Then, the probability of obtaining a time difference of i is the number of squares for which its representative falls in the corresponding region, divided by the total number of squares, which is 2^{2s+2}. To bound the difference in probability, we observe that once more only the tainted squares contribute for a difference in probabilities; in this case, the tainted squares are those that lie in a zone where the integer part of the time changes (in Figure 21 this correspond to the dashed lines). Thus, the absolute difference $|p - q|$ is bounded by the total area of the tainted squares. A simple counting argument reveals that the total number of tainted squares is less than $4 \times 2^{s+1} \times m$, where m is the number of vertical lines (m is bounded by twice the largest value of the time schedule). We also need to add the probability of getting failures when performing algorithm RANDOM which is 2^{-h+1}. Thus, we obtain the desired bound in the difference of probabilities, as $|p - q| < 2^{-h+1} + 8 \times T(k) \times 2^{s+1}/2^{2s+2} = 2^{-h+1} + 2^{-s+2}T(k)$. □

Proposition 11.16. If A is a set decidable in polynomial time by the VBE machine operating with type II protocol with unbounded precision and time tolerance $g \in PF$, then $A \in P/poly$. Moreover, if the time schedule T is exponential, then $A \in BPP//log\star$.

Proof: Let $\mathcal{M}(y)$ be the VBE machine operating with mass y deciding A in polynomial time with protocol $(2, UP, g)$, where $g \in PF$. Let $\gamma \in (0, 1/2)$ bound the probability of failure. Since $\mathcal{M}$ runs in polynomial time, there is a polynomial bn^a bounding all of the following, in the computation of an input word of size n: (a) the number of queries made, (b) the maximum possible size of a query and (c) the largest value taken by the time schedule (that is, $T(k)$, where k is the maximum possible size of a query). Consider the advice function f such that $f(n) = \ell'_1{\downharpoonright}_t\#\ell'_2{\downharpoonright}_t\#\cdots\#\ell'_t{\downharpoonright}_t\#\#r'_t{\downharpoonright}_t\#\cdots\#r'_2{\downharpoonright}_t\#r'_1{\downharpoonright}_t$, where $t = bn^a$ and $t + s$ for a suitable choice of s. That is, $f(n)$ is a non-decreasing sequence of dyadic rationals of size $t + s$ dividing the interval $[0, 1]$ in sub-intervals corresponding to each experimental time. Observe that timeouts occur when the test mass lies in (ℓ'_t, r'_t).

The machine which decides set A in polynomial time, using the function f as advice, simply simulates $\mathcal{M}(y)$ for the same input word of size n and replaces protocol calls with SIMULATE$(2, UP, g)(z_1, z_2, s, h)$ for a suitable choice of h. The difference in probabilities, by Proposition 11.15, is less than $2^{-h+1} + 2^{-s+2}\,t$. To find the suitable values of s and h, observe that this machine induces a probabilistic tree in the same way as $\mathcal{M}(y)$, with depth bounded by t which is polynomial in n. Thus, we take h and s such that $3 \times t \times (2^{-h+1} + 2^{-s+2}\,t) < 1/2 - \gamma$, for example, taking $2^h > 12t/(1/2 - \gamma)$ and $2^s > 24t^2/(1/2 - \gamma)$, so that h and s are logarithmic in n. Then, by Proposition 11.2, the difference in the probabilities of acceptance is bounded by a constant less than $1/2 - \gamma$ and so the probability that this machine gives a wrong answer is bounded by a constant less than $1/2$. Since each simulation can be done in polynomial time, this machine also runs in polynomial time. It follows that $A \in P/poly$.

Finally, in the assumption that T is exponential, the maximum possible size of a query is bounded by some value σ which is logarithmic in n. But in this case, we can take an advice function $\tilde{f}$ consisting on the binary expansion of y from which we can retrieve approximations of ℓ'_i and r'_i by taking $\ell''_i = y\,(i^2 - 1)/(i^2 + 1)$ and $r''_i = y\,(i^2 + 1)/(i^2 - 1)$. It can be seen in a similar reasoning to previous proofs that to get an approximation with precision $2^{-\sigma-s}$, we need $\sigma + s + 2$ digits of y, thus $\tilde{f}$ can be taken to be a prefix function in log. Now, we can retrieve f from $\tilde{f}$ and repeat the same procedure to decide A in polynomial time, thus concluding that $A \in BPP//log\star$. □

For the fixed precision case, all we need to do is devise once more an algorithm to simulate queries and then prove the upper bound.

Proposition 11.17. Let r be any of the following results 'FIRST,' 'SECOND,' 'TIMEOUT' or 'INDISTINGUISHABLE.' If p is the probability of obtaining the result r in the protocol call COMPARE$[2, FP(\epsilon), g](z_1, z_2)$, for an unknown mass y, any time schedule T and any time precision $g \in PF$, and q is the probability of obtaining result r in algorithm SIMULATE$[1, FP(\epsilon), g](z_1, z_2, \sigma, h)$ of Figure 22 receiving as advice dyadic rationals $\ell''_1, \ldots, \ell''_t, r''_t, \ldots, r''_1$ such that $|\ell''_i - \ell'_i|$, $|r''_i - r'_i| < 2^{-\sigma}$ and an approximation of ϵ with error less than $2^{-\sigma}$, then $|p - q| < 2^{-h+2} + \epsilon\, 2^{-\sigma+3} + 2^{-\sigma+3}\, T(|z_1|)/\epsilon$.

```
ALGORITHM "SIMULATE$[2, FP(\epsilon)](z_1, z_2)$"
input two dyadic rational numbers $z_1$ and $z_2$ – both with same size $k$ –
    and natural numbers $\sigma$ and $h$ – precision desired;
Advice consists of a sequence of dyadic rationals $\ell''_1, \cdots, \ell''_T, r''_T, \cdots, r''_1$
    approximating the section numbers and $\epsilon$ (size $\sigma$);
Find $t$ such that $2^{-t-1} < \epsilon \leq 2^{-t}$;  %Just count the number of 0s in the head of $\epsilon$;
    %This can be done with $s + 1$ coin tosses
repeat $h$ times
    Randomly choose a dyadic rational $z'_1$ of size $\sigma$ in $(z_1 - 2^{-t}, z_1 + 2^{-t})$;
        %This can be done with $\sigma - t + 1$ coin tosses
    if $z'_1 \in (z_1 - \epsilon, z_1 + \epsilon)$ then break
end repeat;
If $z'_1 \notin (z_1 - \epsilon, z_1 + \epsilon)$ then return "TIMEOUT";
repeat $h$ times
    Randomly choose a dyadic rational $z'_2$ of size $\sigma$ in $(z_2 - 2^{-t}, z_2 + 2^{-t})$;
        %This can be done with $\sigma - t + 1$ coin tosses
    if $z'_2 \in (z_2 - \epsilon, z_2 + \epsilon)$ then break
end repeat;
if $z'_2 \notin (z_2 - \epsilon, z_2 + \epsilon)$ then return "TIMEOUT";
SIMULATE$(2, IP, g)(z'_1, z'_2, h)$ with advice given by the section numbers
```

Fig. 22. Procedure to simulate an oracle query of size k; receives as advices approximations $\ell'_1, \ldots, \ell'_t$ and $r'_1, \ldots, r'_t$ of the section numbers $\ell_1, \ldots, \ell_t$ and $r_1, \ldots, r_t$ and of the fixed precision ϵ; assume that $g \in PF$.

Proof: There are four situations that change the probability of a given result: (a) the algorithm failed in producing a desired dyadic rational of size σ, (b) the algorithm failed in sampling a random integer in $[-g(z_1), g(z_1)]$, (c) the algorithm does not take into account a small area on the outer part of the region R and (d) the algorithm has different probabilities in the inner part of the region R. The first situation occurs with probability less than $2^{-h} + 2^{-h}$. The second situation occurs also with probability less than $2^{-h} + 2^{-h}$. Regarding the third and fourth situations, let $N = \lfloor \epsilon \times 2^{\sigma} \rfloor > \frac{1}{2}\epsilon 2^{\sigma}$; the small region that is not accounted has area less than $4 \times 2\epsilon \times 2^{-\sigma}$. For the last situation, we repeat the reasoning as in Proposition 11.15; the number of tainted squares is less than $4NT(|z_1|)$. We obtain the desired bound of $2^{-h+2} + \epsilon\, 2^{-\sigma+3} + 2^{-\sigma+3}\; T(|z_1|)/\epsilon$. □

Proposition 11.18. If A is a set decidable in polynomial time by the VBE machine operating with type II protocol with fixed precision ϵ and time precision $g \in PF$, then $A \in BPP//log\star$.

Proof: Let $\mathcal{M}(y)$ be the VBE machine operating with mass y deciding A in polynomial time with protocol $(2, FP(\epsilon), g)$, where $g \in PF$. Let $\gamma \in (0, 1/2)$ bound the probability of failure. Since $\mathcal{M}(y)$ runs in polynomial time, there is a polynomial bn^a bounding the number of queries made (which bounds the depth of the computation tree) and the largest value taken by the time schedule, during any computation of any word of size n. We consider an advice function such that $f(n)$ contains the bits of y and ϵ. For an input word of size n, let $t = bn^a$. First, we take h and σ such that $3 \times t \times (2^{-h+2} + \epsilon\, 2^{-\sigma+3} + 2^{-\sigma+3}\, t)/\epsilon < (1/2 - \gamma)$, which can be achieved with $2^h > 24t/(1/2-\gamma)$ and $2^\sigma > 48t(\epsilon + t/\epsilon)/(1/2-\gamma)$. Once more, h and σ are logarithmic in n. To get approximations of the section numbers with error less than $2^{-\sigma}$, we need $\sigma+2$ bits of y. This means that our advice f will contain the first $\sigma+2$ bits of y and σ bits of ϵ. The advice function can be specified as in previous proofs, that is, at each new power of 2, we append to the advice a constant amount of bits of each of these three constants. Thus, f is a prefix function in log.

The machine that decides set A in polynomial time, using f as advice, begins by using the approximations of y to produce approximations to the section numbers. Then, it simulates $\mathcal{M}(y)$ for the same input word, and also replaces the protocol calls with $\textsc{simulate}(2, FP(\epsilon), g)(z_1, z_2, \sigma, h)$ for the suitable choices of σ and h mentioned above, using as advice the approximations of the section numbers and of ϵ. The difference in probabilities, by Proposition 11.15, is less than $2^{-h+2} + \epsilon\, 2^{-\sigma+3} + 2^{-\sigma+3}\, t/\epsilon$. Then, by Proposition 11.2, the difference in the probabilities of acceptance is bounded by a constant less than $1/2-\gamma$ and so the probability that this machine gives a wrong answer is bounded by a constant less than 1/2. Since each simulation can be done in polynomial time, this machine also runs in polynomial time. It follows that $A \in BPP//log\star$. □

12. Concluding remarks

12.1. *Computational theory of measurement*

Measurement theory is about operations on the real world that define real numbers: see Hempel (1952), Carnap (1966), and the three-volume (Krantz et al. 1990). In our computational approach, we have considered an idealised experimenter as a Turing machine, and an idealised experiment to measure a physical quantity as an oracle to the Turing machine.

Typically, to measure the value of a physical quantity as a real number, we use approximations. In principle, whenever possible, the algorithm conducting the experiment should approximate the unknown quantity by a series of experimental values that converges. The time needed to consult the oracle is not any more a single step of computation but a number of time steps that will depend on the precision. Two-sided measurements have been considered in Beggs et al. (2008, 2010b,c, 2012a,b) and threshold experiments in Beggs et al. (2013a,b). The main complexity classes of Turing machines coupled with these measurements have been studied here for the case of vanishing quantities. We have seen two vanishing experiments, there are more: the Brewster angle experiment is an example of a common physical measurement that falls into this class, while the VBE is constructed to be simple to describe and use in theory development. What is the scope of these results? To answer this we need to:

i. axiomatise a specification of the interface to a physical oracle, and

ii. provide certification lemmas to check whether a given experiment actually satisfies the oracle interface.

This is an open problem. A first attempt is Beggs et al. (2014), which gave a lower bound.

Type of Oracle		Infinite	Unbounded	Finite
Two-sided	lower bound	$P/log\star$	$BPP//log\star$	$BPP//log\star$
	upper bound	$P/poly$	$P/poly$	$P/poly$
	upper bound (w/ exponential T)	−−	−−	−−
Threshold	lower bound	$P/log\star$	$BPP//log\star$	$BPP//log\star$
	upper bound	−−	−−	−−
	upper bound (w/ exponential T)	$P/log\star$	$BPP//log\star$	$BPP//log\star$
Vanishing Type 1 (Parallel)	lower bound	$P/poly$	$P/poly$	$BPP//log\star$
	upper bound	$P/poly$	$P/poly$	$BPP//log\star$
	upper bound (w/ exponential T)	−−	−−	−−
Vanishing Type 2 (Clock)	lower bound	$P/log\star$	$BPP//log\star$	$BPP//log\star$
	upper bound	$P/poly$	$P/poly$	$BPP//log\star$
	upper bound (w/ exponential T)	−−	$BPP//log\star$	−−

Fig. 23. Table of complexity classes of polynomial time Turing machines with different experiments, considered with different concepts of precision and time tolerance. Two-sided experiments have been considered in Beggs et al. (2008, 2009, 2010b,c, 2012a,b) and threshold experiments in Beggs et al. (2013a).

12.2. *Comparing the power of oracles*

The original and primary purpose of an oracle is to boost algorithms. Let us compare the characteristics of the vanishing experiments described here with the other types of experiments described previously – see Beggs et al. (2008, 2010b,c, 2012a,b, 2013b). The three types of experiments have the following common characteristics:

— A *real value* is being measured.

— CHARACTERISTIC 1. A *query* corresponds to a classical query to the oracle in the following sense: the answer is 'YES,' or 'NO,' or 'TIMEOUT' (meaning no answer in the given time).

— *Queries* express dyadic rational putative values of the concept being measured.

— The *cost* of the oracle to the Turing machine expresses the time required by the experiment.[‡‡]

Note that CHARACTERISTIC 1 clarifies that the answer should be a qualitative aspect of the experimental apparatus, e.g., a detection, and not dependent on prior measurements. The threshold oracles introduced in Beggs et al. (2013a,b) have a different CHARACTERISTIC 1, namely:

— CHARACTERISTIC 2. A *query* corresponds to a classical query to the oracle in the following sense: the answer is 'YES,' or 'TIMEOUT.'

The vanishing oracles studied in this paper differ also from CHARACTERISTIC 1, 2, namely:

[‡‡] Remember that the classical connection between the Turing machine and the oracle is a one-step computation.

— CHARACTERISTIC 3. A *query* does not correspond to a classical query to the oracle in the following sense: two queries are made at the same machine (experimenter) time; if the oracle answers first to query number one, then it is interpreted as a 'YES,' else if the oracle answers first to query number two, then is interpreted as a 'NO,' and otherwise a 'TIMEOUT' or an 'INDISTINGUISHABLE' is returned.

Figure 23 reports on lower and upper bounds of VBE machines together with the results of our previous research on the two-sided and threshold oracles.

12.3. *Analogue systems*

There are many forms of computational system involving analogue data. How relevant are our physical oracles to such models and their applications? Most of the systems with real-valued parameters are based on a measurement: part of the control structure of the system reads – in linear time, bit by bit – the binary expansion of some parameter. For instance, this construction is found in the analogue recurrent neural net (ARNN) model of Siegelmann and Sontag (1994), in the optical computer of Woods and Naughton (2005), and in the mirror system of Bournez and Cosnard (1996). In the ARNN case, a subsystem of about 11 neurones performs a measurement of the unique non-rational weight of the network, approximating its value both from above and from below. Once the measurement is done, up to some precision, the computation resumes. Thus, we conclude that:

> *Models of analogue computation 'execute' a measurement assisted to some extent by an algorithm, followed by a computation of arbitrary complexity.*

The theory of analogue systems can then be reduced to algorithms with the ability of making measurements. One way of doing so is by considering the measurement as an oracle consulting algorithm.

This oracle has a cost function $T : \mathbb{N} \to \mathbb{N}$ that gives the number of time steps allowed to perform the measurement of the next bit. The common dynamic systems in the computational literature, having real parameters, perform measurements that cannot be done in linear time. In a balance, the pans move with acceleration that depends on the difference of masses placed in them, in a way such that the time needed to detect a mass difference increases exponentially with the precision of the measurement, no matter how small (yet fixed) that difference can be made. This measurement has an exponential cost that should be considered in the complexity of the decision problem.[§§]

12.4. *Physical systems and real number computation*

The computability of physical systems has been studied from time to time over a long period. Kreisel drew attention to the problem in Kreisel (1974) and stimulated a wave

[§§] In the non-analytic piecewise linear neural net case, the cost function is like the common oracle Turing machine: one-step consultation device, since any further bit has the constant cost of k transitions, for some constant $k \in \mathbb{N}$ – see Siegelmann and Sontag (1994). This is due to the fact that the activation function is piecewise linear instead of the common analytic sigmoid.

of interest, represented by the work of Marion Pour El and Ian Richards, e.g., the papers (Pour-El 1974; Pour-El and Richards 1979, 1981) and the monograph (Pour-El and Richards 1989). The equations of physics constitute a vast subject dominating Mathematical Analysis. The computable analysis of the equations of physics is an essential activity, one that goes hand-in-hand with modelling systems under various assumptions. Modelling requires persistence – Pour El and Richard's striking and influential non-computability results are complemented by computability results in Weihrauch and Zhong (2002). Today's literature on the computability of mathematical models of physical systems is substantial, and the literature on computable analysis, where the tools for this programme are to be found, is very considerable.

However, the approach here is not to start with established equations but with thinking about physical systems. Questions arise from exploring physical situations: how does data picture them, and how are operational aspects of physical systems to be theorised and modelled mathematically? Our interest began with an extreme approach in which we sought to exclude computability theory, attempting to develop algorithmic notions – data and operations – from physical processes. This required a methodology and an interest in physical models, e.g., in Beggs and Tucker (2006, 2007a,b). The combination of computability theory and physical models for physical oracles has enabled us to create a second, new approach to the computability of physical systems that introduces new and influential notions and questions. For example, the simple question, what effect does the act of measurement have on a real number computation? Any physical experiment reveals that using the real number involves:

i. subtle issues of precision;

ii. cost of consultation – e.g., the fact that exponential time on the size of the query is common in Physics;

iii. stochastic behaviour.

The physical oracle answers queries in a time $T : \mathbb{N} \to \mathbb{N}$, dependent upon the size of the query, modelling the fact that successive approximations have a cost that is not necessarily linear in the number of bits of precision obtained.

For those seeking the clarity and stability of a pure mathematical understanding of what are mathematical theorems, the combination – or intrusion – of physical models gets in the way of a pure theory of real number oracles that should abstract from and be applicable to physical measurements. However, the use of physical oracles should be considered a strength of our approach, and necessary at least at this stage of our understanding.

In the case of two-sided experiments, of which we have accumulated many physical examples, our theories changed and grew quite surprisingly. For example, one can contrast the early analysis of the wedge in Beggs and Tucker (2007b) and Beggs et al. (2008) with its subsequent treatment in Beggs et al. (2012b): this was made possible by our experience working on with the collider experiment in Beggs et al. (2010b). Collecting different physical oracles has been hugely productive, and the new classification of experiments in Beggs et al. (2014) comes directly from thinking about and analysing physical experiments.

For the pure mathematician, these experiments may seem to be baggage: easy experiments seem unnecessary and complicated experiments vexing. But for our scientific work they were, and continue to be, essential. Certainly, after collecting many two-sided experiments, we provided a more general axiomatic approach to the computational power theorems (Beggs et al. 2012a). However, that analysis required subtle *certification lemmas* about physical models to enable the general theorems to be applied. In due course, we have no doubt that interesting pure mathematics will emerge, but the time is not right. Indeed, as we have looked at more complicated physical processes, we have found new technical notions and tricks (timing and boundary numbers in this paper being an example). Thus, our program is a work in progress, and, so far, it is true to say that progress can be attributed to engagement with physical examples.

Acknowledgements

The research of Diogo Poças was funded by FCT Doctoral Grant (Portugal).

References

Balcázar, J.L., Días, J. and Gabarró, J. (1990). *Structural Complexity I*, 2nd edition, Springer-Verlag.

Beggs, E., Costa, J.F., Loff, B. and Tucker, J.V. (2008). Computational complexity with experiments as oracles. *Proceedings of the Royal Society, Series A (Mathematical, Physical and Engineering Sciences)* **464** (2098) 2777–2801.

Beggs, E., Costa, J.C., Loff, B. and Tucker, J.V. (2009). Computational complexity with experiments as oracles II. Upper bounds. *Proceedings of the Royal Society, Series A (Mathematical, Physical and Engineering Sciences)* **465** (2105) 1453–1465.

Beggs, E., Costa, J.F., Poças, D. and Tucker, J.V. (2013a). On the power of threshold measurements as oracles. In: Mauri, G., Dennunzio, A., Manzoni, L. and Porreca, A.E. (eds.) *Unconventional Computation and Natural Computation (UCNC 2013)*, Lecture Notes in Computer Science, volume 7956, Springer-Verlag, 6–18.

Beggs, E., Costa, J.F., Poças, D. and Tucker, J.V. (2013b). Oracles that measure thresholds: The turing machine and the broken balance. *Journal of Logic and Computation* **23** (6) 1155–1181.

Beggs, E., Costa, J.F. and Tucker, J.V. (2010a). Computational models of measurement and Hempel's axiomatization. In: Carsetti, A. (ed.) *Causality, Meaningful Complexity and Knowledge Construction*, Theory and Decision Library A, volume 46, Springer-Verlag, 155–184.

Beggs, E., Costa, J.F. and Tucker, J.V. (2010b). Limits to measurement in experiments governed by algorithms. *Mathematical Structures in Computer Science* **20** (06) 1019–1050. Special issue on Quantum Algorithms, editor Salvador Elías Venegas-Andraca.

Beggs, E., Costa, J.F. and Tucker, J.V. (2010c). Physical oracles: The Turing machine and the Wheatstone bridge. *Studia Logica* **95** (1–2) 279–300. Special issue on Contributions of Logic to the Foundations of Physics, editors D. Aerts, S. Smets & J. P. Van Bendegem.

Beggs, E., Costa, J.F. and Tucker, J.V. (2010d). The Turing machine and the uncertainty in the measurement process. In: Guerra, H. (ed.) *Physics and Computation, P&C 2010*, CMATI – Centre for Applied Mathematics and Information Technology, University of Azores, 62–72.

Beggs, E., Costa, J.F. and Tucker, J.V. (2012a). Axiomatising physical experiments as oracles to algorithms. *Philosophical Transactions of the Royal Society, Series A (Mathematical, Physical and Engineering Sciences)* **370** (12) 3359–3384.

Beggs, E., Costa, J.F. and Tucker, J.V. (2012b). The impact of models of a physical oracle on computational power. *Mathematical Structures in Computer Science* **22** (5) 853–879. Special issue on Computability of the Physical, editors Cristian S. Calude and S. Barry Cooper.

Beggs, E., Costa, J.F. and Tucker, J.V. (2014). Three forms of physical measurement and their computability. *Reviews of Symbolic Logic* **7** (4) 618–646.

Beggs, E. and Tucker, J.V. (2006). Embedding infinitely parallel computation in Newtonian kinematics. *Applied Mathematics and Computation* **178** (1) 25–43.

Beggs, E. and Tucker, J.V. (2007a). Can Newtonian systems, bounded in space, time, mass and energy compute all functions? *Theoretical Computer Science* **371** (1) 4–19.

Beggs, E. and Tucker, J.V. (2007b). Experimental computation of real numbers by Newtonian machines. *Proceedings of the Royal Society, Series A (Mathematical, Physical and Engineering Sciences)* **463** (2082) 1541–1561.

Bekey, G.A. and Karplus, W.J. (1968). *Hybrid Computation*, John Wiley & Sons.

Born, M. and Wolf, E. (1964). *Principles of Optics. Electromagnetic Theory of Propagation, Interference and Diffraction of Light*, second (revised) edition, Pergamon Press.

Bournez, O. and Cosnard, M. (1996). On the computational power of dynamical systems and hybrid systems. *Theoretical Computer Science* **168** (2) 417–459.

Carnap, R. (1966). *Philosophical Foundations of Physics*, Basic Books.

Geroch, R. and Hartle, J.B. (1986). Computability and physical theories. *Foundations of Physics* **16** (6) 533–550.

Hempel, C.G. (1952). Fundamentals of concept formation in empirical science. *International Encyclopedia of Unified Science*, volume 2 no. 7, Chicago Univ. Press.

Krantz, D.H., Suppes, P., Duncan Luce, R. and Tversky, A. (1990). *Foundations of Measurement*, vol. 1 (1971), vol. 2 (1989) and vol. 3 (1990), Academic Press.

Kreisel, G. (1974). A notion of mechanistic theory. *Synthese* **29** (1) 11–26.

Pauly, A. (2009). Representing measurement results. *Journal of Universal Computer Science* **15** (6) 1280–1300.

Pauly, A. and Ziegler, M. (2013). Relative computability and uniform continuity of relations. *Journal of Logic and Analysis* **5** (7) 139.

Pour-El, M. (1974). Abstract computability and its relations to the general purpose analog computer. *Transactions of the American Mathematical Society* **199** 1–28.

Pour-El, M. and Richards, I. (1979). A computable ordinary differential equation which possesses no computable solution. *Annals of Mathematical Logic* **17** (1–2) 61–90.

Pour-El, M. and Richards, I. (1981). The wave equation with computable initial data such that its unique solution is not computable. *Advances in Mathematics* **39** (4) 215–239.

Pour-El, M. and Richards, I. (1989). *Computability in Analysis and Physics*, Perspectives in Mathematical Logic, Springer-Verlag.

Siegelmann, H.T. and Sontag, E.D. (1994). Analog computation via neural networks. *Theoretical Computer Science* **131** (2) 331–360.

Weihrauch, K. (2000). *Computable Analysis*, Springer-Verlag.

Weihrauch, K. and Zhong, N. (2002). Is wave propagation computable or can wave computers beat the Turing machine? *Proceedings of the London Mathematical Society* **85** (2) 312–332.

Woods, D. and Naughton, T.J. (2005). An optical model of computation. *Theoretical Computer Science* **334** (1–3) 227–258.

Ziegler, M. (2009). Physically-relativized Church-Turing hypotheses: Physical foundations of computing and complexity theory of computational physics. *Applied Mathematics and Computation* **215** (4) 1431–1447.